Confirming a user's identity by means of a biometric based on their typing behaviour

by

Thato Mokoena

A dissertation submitted in partial fulfillment of the requirements to the study towards the degree of

Master of Engineering in Electrical and Electronic Engineering

in the

Faculty of Engineering and the Built Environment

at the

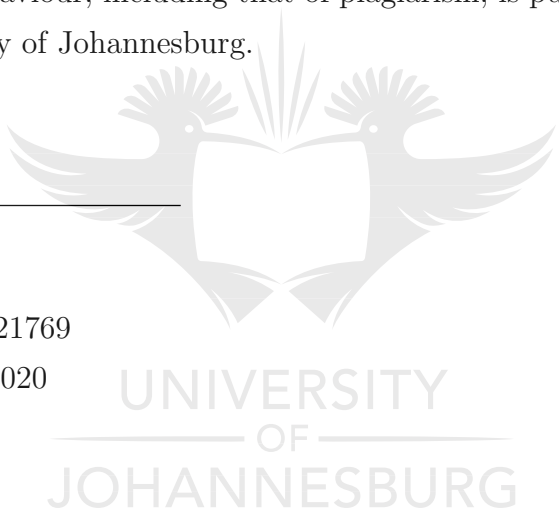
University of Johannesburg

August, 2020

Supervisor: Dr. D.G. Sabatta

Plagiarism Declaration

I, **Thato Mokoena**, hereby declare that the content within this document is of my own work and has not been submitted anywhere else for academic credit, either by myself or another person. I understand what plagiarism constitutes and hereby declare that this mini-dissertation document is of my own ideas, words, arguments, graphics, and results, except where reference is explicitly made to another person's work. I understand that any unethical academic behaviour, including that of plagiarism, is punishable by disciplinary action by the University of Johannesburg.



Thato Mokoena
Student Number: 201421769
Tuesday 18th August, 2020

"Kung fu. It means, 'supreme skill from hard work.' A great poet has reached kung fu. The painter, the calligrapher, they can be said to have kung fu. Even the cook, the one who sweeps steps or a masterful servant can have kung fu. Practice. Preparation. Endless repetition. Until your mind is weary, and your bones ache. Until you're too tired to sweat, too wasted to breathe. That is the way, the only way one acquires kung fu."

Marco Polo, Season 1, Episode 3: Feast

Acknowledgements

I would like to thank the useful contributions made by my supervisor, Dr. D.G. Sabatta throughout the duration of this dissertation.

I would also like to give much thanks to the students of the University of Johannesburg who participated in the study by providing their typing data samples.

I am also grateful to Telkom SA via the Telkom Centre of Excellence (CoE) at the University of Johannesburg for their financial support during the course of the study.

Lastly, I wish to express my sincere gratitude to my family and friends for the love, encouragement and support.

Abstract

Conventional authentication methods used in computer systems such as Personal Identification Number (PIN), password and token present security flaws (or vulnerabilities) as they can be guessed, cracked, stolen or shared. Biometric authentication schemes have presented themselves as better options as compared to traditional authentication methods as they exploit the uniqueness of a subject with regards to what they are or how they behave.

This dissertation deals with the use of a behavioural biometric, Keystroke Dynamics as a means of authentication for verifying online users. Firstly, we present a method that reduces the amount of data by removing common typing patterns. We then show that eliminating common patterns improves the discriminating ability of a classifier while reducing the amount of data and therefore the time required to train a classifier.

We also present a novel keystroke dynamics authentication method that is based on text retrieval concepts and methods. For the evaluation of our algorithm, we collect two datasets in real-life environments. We then use our datasets together with other datasets found in literature to test our algorithm on both classification and authentication-based tasks. Furthermore, we show experimentally that a person's typing behaviour is susceptible to the environment in which they are typing.

Contents

List of Figures

List of Tables

Abbreviations

2FA Two Factor Authentication.

ATM Automated Teller Machine.

CER Crossover Error Rate.

CMS Course Management System.

CSV Comma-Separated Values.

EER Equal Error Rate.

FAR False Alarm Rate.

FMR False Match Rate.

FN False Negative.

FNMR False Non-Match Rate.

FP False Positive.

FRR False Rejection Rate.

HTML Hypertext Markup Language.

IE Internet Explorer.

IPR Impostor Pass Rate.

JS JavaScript.

KSD Keystroke Dynamics.

ML Machine Learning.

PIN Personal Identification Number.

ROC Receiver Operating Characteristic.

SaaS Software as a Service.

SMS Short Message Service.

VPL Virtual Programming Lab.

Glossary of Important Terms

Biometrics The science of using human physical, chemical and behavioural attributes to uniquely identify a person.

Biometric System A pattern recognition system which makes a personal identification or verification by determining the authenticity of a specific physiological or behavioural characteristic possessed by the subject.

Behavioural Biometrics A branch of biometrics focused on measuring patterns in how humans perform activities.

Keystroke Dynamics A type of a behavioural biometric based on the assumption that individuals have unique typing rhythms.

Key Event A single action with a keyboard key. It is a key press if the key is pressed, or a key release if the key is let go.

Key Logging The act of recording key events.

Keystroke The press and release of a single key.

Continuous Authentication A continuous authentication approach that verifies the user's identity based on the user's ongoing interactions with the system.

Chapter 1

Introduction

Over the years, a lot of time and effort have been invested in improving and enhancing security and privacy of user accounts on various online platforms. However, throughout history, we have learnt that there is no such thing as a perfect security system because hackers find a loophole in a system. A lot of research and implementation of various static and dynamic authentication methods has been done where a number of authentication techniques such as the traditional PIN and password, biometrics and modern approaches such as two factor authentication are combined to produce robust security systems [1]. Biometric verification refers to the process of validating a subject's identity by means of traits and characteristics that are unique to an individual such as fingerprint, face, signature and voice, to mention but a few [1].

Educational institutions (mostly institutions of higher education) have adopted and implemented the concept of e-learning and drifted away from doing every teaching and learning activity through the conventional classroom approach [2]. This brought a large increase in online learning courses offered by universities and other institutions via platforms like edX, Udacity, Coursera and Udemy. Platforms such as Moodle have been built to make publishing online courses much easier. Moodle reports just above 90,000 registered sites in over 200 countries around the world offering more than fourteen million courses [3]. With this development came great benefits such as improved access to education, affordable quality education, distance learning and flexibility when considering issues of place and time [4, 2]. The biggest challenge however with e-learning and other online platforms is the issue of confirming the identity of the individual engaging in the assessments. This challenge poses a threat to the credibility of online assessments and courses. The posed question is: *Whether the person performing online educational activities is the registered student?*

Conventional authentication methods such as PIN, password and token have become less robust over the years as they can be guessed, cracked, stolen or shared [1]. This has created the need for more robust methods that can exploit the uniqueness of every user to enhance security and address the online identity crisis. Apart from the application of biometrics in this study, biometrics have positioned themselves as the more effective solution for human recognition in applications such as secure access control, law enforcement and international border crossing. Biometrics, such as facial recognition and fingerprints can be employed in the attempt of solving this problem since they provide a more personalized security since the way each individual types is unique. Biometrics have proved to be more effective as they are more difficult to spoof due to their uniqueness property and they cannot be shared.

This dissertation investigates the use of Keystroke Dynamics (KSD) as a means of verifying a user's identity online. KSD is a type of behavioural biometric used to monitor a user's typing signature by extracting and analysing timing information while they interact with a keyboard in an attempt to identify them based on the rhythmic patterns in how they type [5]. This biometric technology is employed in computer security, especially on online platforms, due to the inherent anonymity of being on the Internet. The use cases include, verification of online test takers [6, 7], password strengthening [5] and in online banking. Some of the factors that make KSD attractive as a means of authentication are [8, 9]:

- No additional sensor is required as an existing keyboard is used to acquire keystroke data.
- Less obtrusive, since the user activities are not affected as data is acquired while the user is busy with their online activities.
- Keystroke dynamics are available after the initial authentication step at the start of an online session so they can be used to continuously verify the user.

KSD can be used together with traditional methods like PIN-Password as a supplement in order to enhance the robustness of these conventional security. The benefit with biometrics is that it provides personalized security which is not easy to crack.

1.1 Problem Statement

E-learning is becoming more and more popular as a way to increase the availability of education to a larger audience. The use of e-learning has simplified and improved the

process of teaching and learning. However, a big problem in e-learning is the question of verifying the identity of the person who is taking an online assessment to confirm that the user is the registered student and not someone else. This identity verification question needs to be addressed to ensure the credibility of electronic assessments and qualifications obtained via online courses. At present, conventional authentication methods such as PIN, passwords, tokens and Short Message Service (SMS) based authentication are being used. The challenge with the aforementioned methods is that they can be cracked, stolen or the information can even be shared willingly by the rightful owner.

1.2 Research Question

To what degree of accuracy can the identity of a user be confirmed using a biometric based on the user's typing signature?

1.3 Research Objectives

The primary objective of this study shall be to design, implement and test a system for verifying the identity of a website user based on the users typing signature. The system shall enrol users and verify their identity at a later stage. Furthermore, the proposed solution shall detect and report anomalies. The secondary objectives of this dissertation are:

1. To identify optimum features and their possible combinations that uniquely represent the user's typing behaviour in a way that ensures high accuracy of authentication.

2. To investigate whether a user's typing behaviour is consistent regardless of the change in environments.

3. To implement a classification technique that will ensure a high degree of accuracy.

1.4 Research Scope

This project will focus on the design and implementation of algorithms that attempt to accurately confirm the identity of a website user based on the user's typing biometric. This study will be subject to the following constraints:

- The study will be limited to the application of keystroke dynamics as a method of verification or confirmation and not identification.

- The study will be limited to the e-learning domain.
- The free-text enrolment strategy will be used.
- The target device in this study will be a desktop Personal Computer (PC) or a laptop. Therefore, mobile phones, tablet PCs or any devices with virtual keyboards will be excluded.
- Only the QWERTY computer keyboard will be used as a sensor.

1.5 Contributions

This dissertation makes the following contributions:

1. Two datasets of anonymous users and their typing patterns collected in real-world situations. The datasets are collected in two different environments, namely, a closed and an open environment.
2. Two methods of detecting and removing common typing traits amongst users in a typist dataset. The first method is based on a machine learning novelty detection algorithm, the One-Class Support Vector Machines (OC-SVM). The second method is based on text retrieval concepts and uses a stop list (list of all words/terms that are used excessively in a language) to filter out typing traits exhibited by most users. This removal of the common data is done with aim of improving the discrimination ability of the classifier.
3. A novel keystroke analysis authentication algorithm for free-text input. This algorithm uses text retrival methods and concepts to convert authentication by keystroke dynamics into a text retirval problem.

1.6 Research Methodology

The methodology applied in this research is founded on the engineering research method and will commence by exploring the existing literature in biometrics authentication particularly keystroke dynamics as an authentication technique. The literature review shall help provide in-depth knowledge regarding the use of keystroke dynamics as a method of authentication in web systems and the current state-of-the-art techniques which are employed to accurately verify the identity of a website user. Furthermore, undertaking the literature study will help investigate metrics that could be employed to evaluate the performance of the implemented algorithms.

The second phase will be focused on data collection and analysis. Here, a web-based keystroke logger that records timing data as subjects interact with the keyboard will be built. The data will then be collected by running the data collection system in a browser on a PC with a QWERTY keyboard. The collected data will then be analysed by employing statistical measures and classifiers in order to select optimal features that could be used for uniquely identifying subjects.

After the all the data has been proccessed to produce the final dataset, the dataset will be split into three different smaller sets, namely, the training, validation and test sets. The training set will be used to train the implemented classification algorithms. To compare variations of the classification models against each other during the experimentation phase, a validation dataset will be used. After the best performing model has been determined, the test dataset will then be used to evaulate the performance of the final model. All of this is done to ensure that the classifications models developed do not overfit.

From performing a review of the achieved results, conclusions will be drawn and recommendations will be suggested for future research.

1.7 Publications

The publications produced during the course of this study include:

- Mokoena, Thato and Sabatta, Deon. Identifying and Removing Common Behaviour in Keystroke Dynamics to Improve Classification. In: Southern Africa Telecommunication Networks and Applications Conference (SATNAC) 2019– The Changing Face of Telcos in a Digital World. 01-04 September 2019, Ballito, KwaZulu-Natal, South Africa. Discussed in Chapter 5.

- Mokoena, Thato and Sabatta, Deon. User Classification by Keystroke Dynamics using Text Retrieval Methods. In: 2020 Joint SAUPEC/RobMech/PRASA Symposium. 29-31 January 2020, Observatory, Cape Town, Western Cape, South Africa. Discussed in Chapters 6 and 7.

1.8 Dissertation Outline

The remainder of this dissertation document is organised as follows:

- **Chapter 2** will give an overview of biometrics, and introduce core concepts that will be used to describe keystroke authentication in subsequent chapters. Furthermore, the existing literature in keystroke dynamics will be explored which includes the state-of-the-art, as well as the similar works that have been done by various research groups.
- **Chapter 3** will provide a detailed description of the data collection process.
- **Chapter 4** will describe how the collected data was analysed and processed in order to extract features and eventually create the dataset used in this study. The analysis will mainly focus on how the data was prepared for usage to be consumed by the algorithms which will be used to learn the user's typing pattern.
- **Chapter 5** will present a method for improving the discrimination ability of a classifier by removing common typing patterns.
- **Chapter 6** will present a novel KSD authentication algorithm for free-text input.
- **Chapter 7** will present the results obtained from evaluating the authentication algorithm proposed in Chapter 6 together with the discussion and analysis of these results.
- **Chapter 8** will provide the conclusion(s) of the study together with the future work and recommendations.

Chapter 2

Background

The key events that formed the basis for keystroke dynamics as we know it today took place in the mid and late 1800s through two inventions and the introduction of the theory of "anthropometrics" by Alphonse Bertillon [10]. The inventions are, the telegraph which was developed in the 1830s by Samuel F.B. Morse and the modern typewriter by Christopher L. Sholes in the 1870s [11, 12, 13]. Morse's telegraph, which was a major long distance communitaction device in the 19th century used the Morse Code, which is a representation of alphabets and numbers as dots and dashes [12, 13]. Typewriters had been around since the 1700s but their keyboards were a single line of keys arranged alphabetically in a piano-like format as shown in Figure 2.1 [14].

Figure 2.1: Single line typewriter keyboard layout [14].

The QWERTY keyboard arrangement (shown in Figure 2.2) was created by Sholes with the aim of preventing neighbouring typewriting bars from jamming by placing common letter pairs onto separate hands [11]. The QWERTY keyboard arrangement is the most used keyboard format today[1]. The most used style of typing during the days of the single line keyboard was the "hunt-and-peck"–first the typist hand to find the keys and then strike them using the index fingers of either hand [11]. When the QWERTY keyboard was introduced typists could not type as fast as they did due to the changes in the format. To remedy this problem touch typing was employed, the typists had to memorize the keys

[1]There is also the Dvorak's keyboard arrangement which is the second most used keyboard layout.

and type using four fingers of each hand, with the thumb used for the space bar. Lessons and training were then given to teach the skill of touch typing.

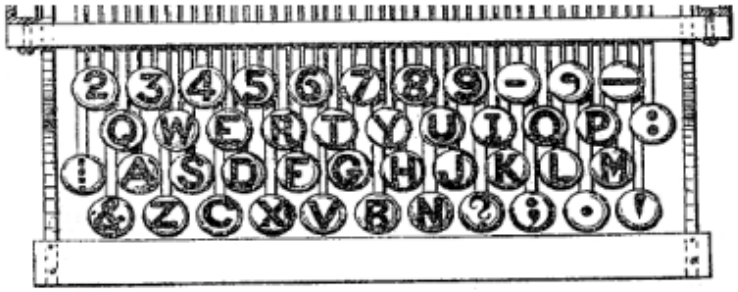


Figure 2.2: Sholes' QWERTY keyboard layout [15].

Dvorak *et al.* [16] were the first to publish a book based on the study of typing behaviour and in this book they made an observation that despite the typists receiving the same lessons and training, each typist had their own style that uniquely discriminates them from the others. It is in this book, were the first example of typing being used a biometric was document. Furthermore, around the time [16] made the aforementioned observations it is said that during telegraph era (early 1900s) telegraph operators could distinguish each other by listening to the tapping rhythm of the dots and dashes [8].

The use of biometrics as a means of verification predates the invention of typing. In ancient Babylon and China fingerprints were used to sign legal documents and in the ancient Egypt the Nile Valley traders were formally identified using physical attributes such as eye colour [17, 10]. Ironically, the field of biometrics was only formally described after the creation of the modern typewriter. It was in 1882 when Bertillon introduced "anthropometrics" ("human measurements") where he took measurements of persons and noting unique features such as scars or tattoos with the aim of using those recorded features to later identify the persons [10]. The same idea introduced by Bertillon came to be known as biometrics ("life measurements") years later. In present time, biometrics are employed in security systems to verify the persons based on the physical and behavioural characteristics such as fingerprints and typing signature respectively.

In the past, the telegraph key and typewriter keyboard served as input devices and today, the computer keyboard, mobile keypad and touchscreen panels serve as input devices. Ever since [18] presented keystroke dynamics as a viable means of authentication in computer security this study field has gained momentum.

The benefits and shortcomings of KSD as a means of verification will be discussed. Furthermore, the existing literature will be explored and presented. This chapter will end with the review of the application of machine learning based classifiers in KSD.

2.1 User Authentication in Computer Security

Authentication is the process of verifying if someone is, in fact, who she/he claims to be. Verification and identification are usually confused and thought to be the same but they are different. Identification involves recognizing a subject by searching through all the stored user templates in the database in order to find a match whereas verification validates a subject's identity by comparing the captured data with the template(s) stored in the system's database belonging to that particular subject [19]. Thus, identification is a $1 : n$ (one-to-many) relationship and verification is a $1 : 1$ (one-to-one) relationship.

In computer security authentication can be achieved by satisfying one or a combination of the three criteria. The criteria are [19, 20]:

- Knowledge based: What you know
- Possession based: What you have
- Biometric based: What you are

An authentication system consists of the following core elements [9, 21]:

- Initiator: a user that needs to be authenticated in order to gain access.
- Distinctive traits: attributes that can uniquely discriminate between different users or groups based on what the user knows, has or is.
- Administrator: responsible for usage of the authentication system and relies of on automatic authentication discriminate authorized users from the unauthorized ones.
- Authentication mechanism: a way of verifying the user or group of users based on the differentiating characteristics such as knowledge factors, possession factors and biometric factors.
- Privilege: given when the user is successfully authenticated by the authentication mechanism, and the same mechanism denies the privilege if the user authentication is unsuccessful.

In the subsections to follow a brief description of each of the authentication criterion is provided with an exception on the biometric based authentication scheme which will be more lengthy as this study is focused on using a type of biometric as means of authentication.

2.1.1 Knowledge Based

For knowledge based authentication one has to remember a secret which is in the form of a PIN, password or passphrase. The advantages of this method include easy implementation, low cost and fast authentication [20]. The challenge however is that a secret may be forgotten when not written, stolen when written, shared when it is convenient or guessed if it is too easy. All these shortcomings increase the limitations of using what one knows as a principle authentication scheme.

2.1.2 Possession Based

In this scheme a physical token is used for means of authentication. The token can be in the form of an electronic keycard, smart card or keys. Unlike the previous technique, this technique does not require any memorizing. However, a token can suffer the same fate as a secret as it can be stolen or shared. Furthermore, the cost of implementing such a system is higher as it would require not only the physical token but the processing equipment for purposes of verification [22].

2.1.3 Biometric Based

This authentication scheme is based on what the subject is. Here, biometrics are employed in order to uniquely identify subjects. The term biometrics is derived from two Greek terms, *bios* which means life and *metrons* which means to measure [17]. Biometrics is the science of using human physical, chemical and behavioural attributes to uniquely identify a person [19]. These attributes include face, fingerprint, voice, keystroke and many others to name but a few. There are two main classes of biometrics namely, physiological/physical (static biometrics) and behavioural (dynamic biometrics) [17, 20]. Physiological biometrics measure a specific part of the structure or shape of a portion of a subjects body [17]. Whereas, behavioural biometrics are more focused on how one does something.

The advantage of employing physical biometrics is that they provide a higher accuracy as compared to behavioural biometrics [1, 23, 24]. However, physical biometric sensor devices have to be implemented and this increases the cost of implementation which is

a limitation. Behavioural biometrics serve as one of the popular techniques in continuous authentication of persons, but it yields inadequate accuracy due to the fact that behaviour is not stable and it changes over time [24, 9]. Figure 2.3 shows the two classes of biometrics and their common examples.

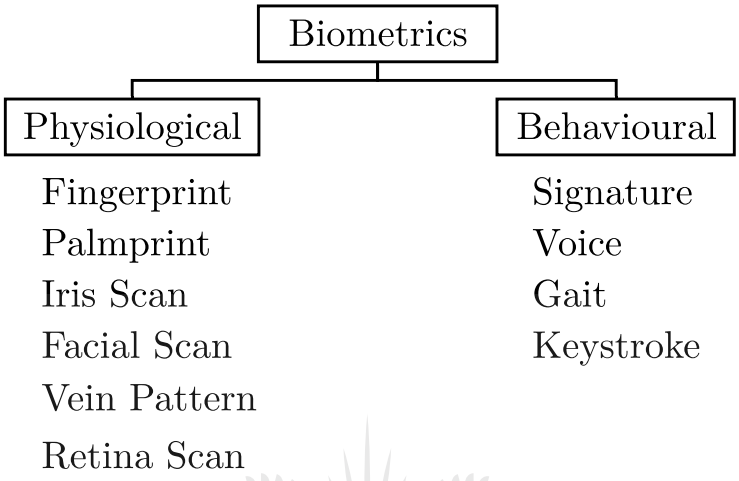


Figure 2.3: The two classes of biometrics and their examples.

2.1.3.1 Operation of a Biometric System

A biometric system is in essence a pattern recognition system that employs a sensor to obtain biometric data from a subject, extracts the main features from the data, compares the obtained feature set against the reference feature set(s) stored in the database, and makes a decision based on the outcome of the comparison [19]. Thus, the operation of a biometric authentication system consists of four main stages [19, 9]:

- Data acquisition: Capturing the user's biometric data from the sensor module. Used during user enrollment and also during verification or identification.
- Feature extraction: Extracting information from the raw biometric data. This done by having an optimum feature set that can uniquely represent users.
- Matching: Comparison between the feature set provided during an authentication session with the biometric template that was constructed during enrollment.
- Decision rule: A rule or set of rules that take into account the outcome of the matching phase and determine whether the user is legitimate or not.

This system works in two phases: enrollment; and verification or identification. During enrollment, the user's biometric template is stored in the database for future reference and during verification the provided user template is compared against the reference template in the system's database. It is required that the user provides their biographical information which will be used to link the user with his/her biometric data [19, 17, 25]. Figure 2.4 shows functional block diagrams of both enrollment and verification processes and how the aforementioned core modules of a biometric authentication interact with each other.

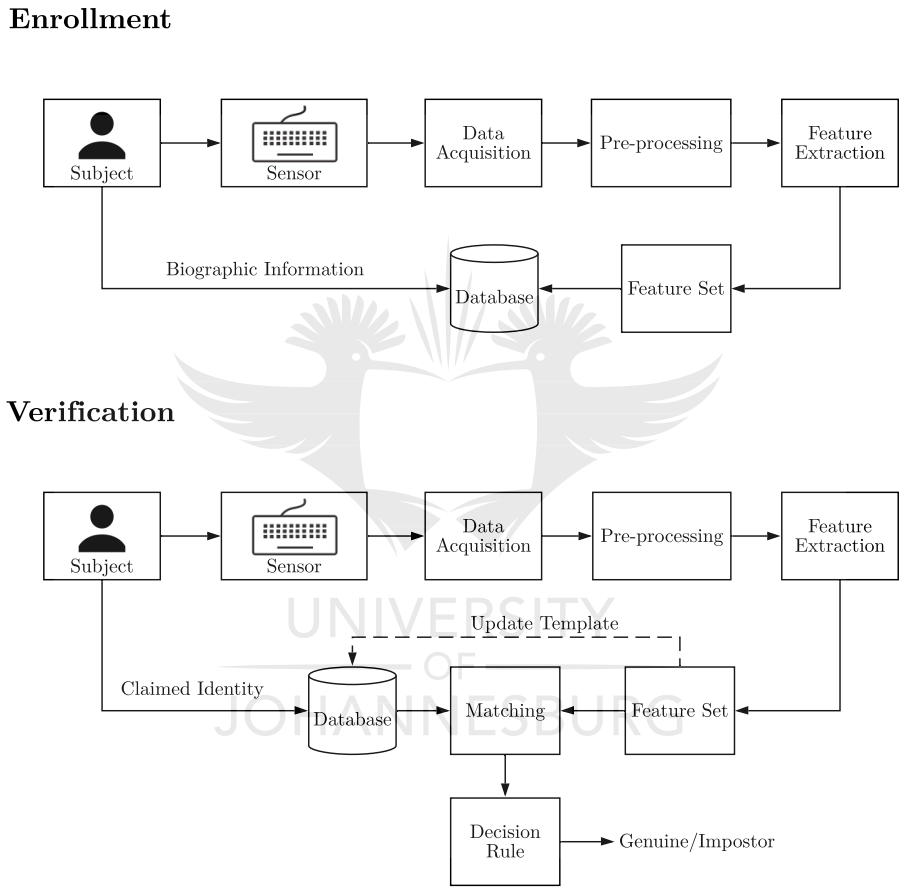


Figure 2.4: Functional block diagrams of the enrollment and verification phases of a biometric system.

2.1.3.2 Properties of a Biometric System

Jain *et al.* [26] state seven properties that determine the suitability of a physical or behavioural characteristic to be used as a biometric measure in a biometric application:

1. Universality: Everyone should possess the biometric measure.

2. Uniqueness: The measure should discriminate between two people which implies that no two people can have the same biometric feature set.

3. Permanence: The trait should be invariant over a period time with respect to the matching algorithm. However, behavioural biometrics change slightly with time.

4. Collectability: The data collection process should be quantitatively measureable.

5. Performance: The biometric system should be accurate to some degree of accuracy that is acceptable.

6. Acceptability: The target population should be willing to accept the measure and share their biometric data.

7. Circumvention: The measure should not be easily imitated or mimicked in case of physical or behavioural traits respectively.

2.1.3.3 Performance of a Biometric System

Unlike PIN-Password based systems, where there should be an exact match between the stored PIN or password and the provided numeric or alphanumeric string in order to validate a subject's identity, a biometric system rarely encounters two samples of a subject's biometric trait that result in exactly the same feature set [19, 17]. This is caused by a number of factors, such as imperfect sensing conditions (e.g., noisy biometric data due to sensor malfunction), changes in the subject's biometric trait (e.g., a person's typing rythym changes with time) and variations in how the subject interacts with the sensor device (e.g., partial fingerprints) [19].

The performance of a biometric system is evaluated based on the errors committed by the system. Two types of error are at the core of evaluating the effectiveness of a biometric system: Type I and Type II which are False Positive (FP) and False Negative (FN) respectively. A Type I error is observed when an impostor is wrongly classified as a genuine user and a Type II error occurs when genuine user is denied access [19, 8, 27]. Using the FP and FN a number of performance metrics can be computed with the aim of giving a clear picture of how well a biometric system can execute task of identification or verification. These evaluation metrics are summarized as follows:

1. False Acceptance Rate (FAR): Percentage ratio between falsely rejected genuine subjects against the total number of genuine subjects in the system [19, 27]. This indicator is based on the Type II error. Sometimes referred to as False Non-Match Rate (FNMR) or False Rejection Rate (FRR) [19].

2. Impostor Pass Rate (IPR): Percentage ratio between falsely accepted unauthorized subjects against the total number of imposters attempting to access the system [27]. This indicator is based on the Type I error. In some literature the Impostor Pass Rate (IPR) is referred to as False Match Rate (FMR) or the False Acceptance Rate (FAR)[2] [19, 27].

3. Equal Error Rate (EER): The value at which the False Alarm Rate (FAR) is equal to the IPR when FAR and IPR are plotted on the same plot. This metric is used to determine the overall accuracy of the system and as a performance measure to compare biometric systems [8, 27]. In other literature this indicator is stated as Crossover Error Rate (CER) [25].

4. Receiver Operating Characteristic (ROC) Curve: A plot of FAR against IPR on a linear, logarithmic or semi-logarithmic scale [19]. The FAR, IPR and EER values can be read obtained directly from this plot [27].

In an ideal authentication system, the FAR and IPR are both equal to zero. In the real world, these errors occur and they have a trade-off; a low FAR implies that less rejection and easy access for the genuine user, however, this means that impostors can also gain access easily provided the can mimic a genuine user to an acceptable degree. To remedy this, the FAR can be reduced so that imposters do not gain easy access, however, this will lead to genuine users being rejected. Thus FAR and IPR have to be selected in a way the prohibits unauthorized access but does not reject authorized users. Hempstalk [27] argues that it is more acceptable to have a security system that rejects genuine users every now and then than one which grants impostors easy access.

2.1.4 Combination of Authentication Schemes

The aforementioned authentication schemes can be combined in order to increase the security of systems thus giving rise to multi-factor authentication security systems. In most cases, biometrics are used to supplement knowledge and posession based authentican schemes in order to add an addtional level of security [19]. The most common example is the combination of two schemes which is known as Two Factor Authentication (2FA) [19]. Examples of 2FA include an Automated Teller Machine (ATM) which requires a PIN (what the subject knows) and a bank card (what the subject possess) for a user to perform transactions. More robust ATMs have been implemented which use all the the three criterion mentioned above by adding a biometric recognition feature by either

[2]To avoid confusion and for consistency sake we will use FAR to only refer to the False Alarm Rate and not the False Acceptance Rate.

face, fingerprint or retina (what the subject is) [28, 29]. Combining these authentication factors results in more robust security systems and thus making it hard to compromise the security of systems that employ mutli-factor authentication techniques.

2.2 Keystroke Dynamics

2.2.1 Features

While typing, the computer can record the key pressed, the time it was pressed and the the time at which it was released. From this raw typing data, KSD timing features can be extracted as shown in Figure 2.5.

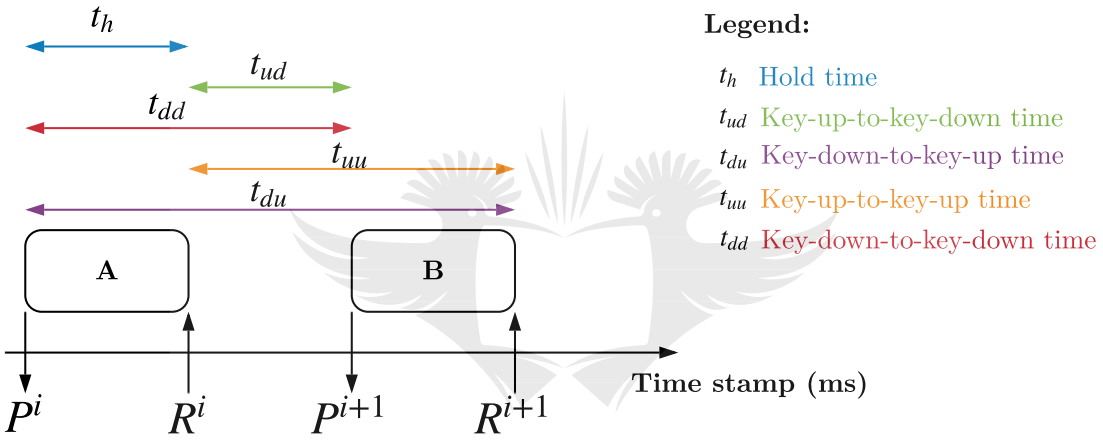


Figure 2.5: Timing features that can be extracted from two successive keystrokes, 'A' and 'B' where P^i and R^i denote the key press and key release time stamps of the first and second keys respectively. The superscript i denotes the position of a keystroke in the sequence of keys typed in from the input stream.

1. Hold time (t_h): The time between pressing and releasing a key. In other words, how long was the key held. It sometimes referred to as *key-down time*, *dwell time* or *key duration*. The *hold time* can be calculated as

$$t_h^i = R^i - P^i \tag{2.1}$$

2. Key-up-to-key-down time (t_{ud})[3]: The time between the key press event of the second key and the key release of the first key. The *key-up-to-key-down time* can be

[3] t_{ud} and t_{uu} may be negative if the key release events of the earlier key occur after the key press and key release events of the successive key. Examples include the use of modifier keys such as `Shift` and `Control`.

calculated as

$$t^i_{ud} = P^{i+1} - R^i \tag{2.2}$$

3. Key-down-to-key-down time (t_{dd}): The time difference between the key press of the second key and the key press of the first key. The *key-down-to-key-down time* can be calculated as

$$t^i_{dd} = P^{i+1} - P^i \tag{2.3}$$

4. Key-up-to-key-up time (t_{uu})[3]: The time difference between release of the second key and the release of the first key. The *key-up-to-key-up time* can be calculated as

$$t^i_{uu} = R^{i+1} - R^i \tag{2.4}$$

5. Key-down-to-key-up time (t_{du}): The time difference between the second key release and the key press of the first key. The *key-down-to-key-up time* can be calculated as

$$t^i_{du} = R^{i+1} - P^i \tag{2.5}$$

In addition to the aforementioned timing features, two keycode based features were employed, fk^i and tk^j; where fk^i is the keycode of the current key and tk^j is the keycode of the successive key. These keycodes are related to the timing features. fk^i is tied with t_h, using these two features one can deduce how long a particular key was held down. fk^i and tk^j are related to the inter-key timing features, using this set of features one can determine the subject's travel times between two keys with fk^i as the initial point and tk^j as the destination.

Teh *et al.* [8] refer to features 2 through 5 (inter-key times) as flight times of different types, namely, type 1 through type 4 respectively. According to the survey of keystroke dynamics biometrics by [8], the most used features are flight times at 49% followed by hold time at 41% and pressure and other features take up the remaining 10%. There exists other features, although uncommon, such as keystroke pressure, typing error frequency, sound of typing and size of the finger which have been used [30, 31].

2.2.2 Benefits and Challenges

Every class of biometrics has its advantages and disadvantages. It can be argued that biometric measures based on physical characteristics have fewer challenges as compared to behavioural biometrics. In this subsection the benefits and challenges faced when

working with free-text keystroke dynamics will be discussed with reference to some of the biometric properties mentioned in Section 2.1.3.2. Some of the advantages of include [8, 22]:

- Biometric scheme is inexpensive as compared to other biometrics since an existing computer keyboard is used as an input device thus there are no additional costs for the sensor device.
- Less obtrusive and does not disturb the user while he/she is performing tasks.
- Offer both static and continuous authentication. The ability of KSD to provide continuous authentication without continuously prompting the user to provide authentication data gives it an upper hand on other biometric schemes.
- A keystroke event is measured with *ms* precision which provides fine granularity thereby yielding a high degree of uniqueness. This makes it impossible for two subjects to have the same template which limits mimicry thus resulting in high uniqueness and high circumvention.

The disadvantages of working with keystroke analysis include [8, 22]:

- Behavioural biometrics change over time as one's behaviour cannot be exactly the same at all times, thus this scheme has low permanence. Due to the changing behaviour the user template may have to be updated regularly.
- User's typing signature may be affected by the user's mental and physical conditions (mood, fatigue, physical injuries on the hands).
- The typing pattern may be influenced by the hardware. For example, typing on desktop keyboard may yield different results compared to typing on a laptop keyboard.

2.3 Conclusion

This chapter introduced the basic theory and background of the field of biometrics and user authentication in computer security with an emphasis on keystroke dynamics. An overview of how biometric systems work was presented together with the metrics used to evaluate the performance of biometrics-based security systems. Lastly, the fundamental features used in keystroke dynamics and the benefits and challenges associated with the use of this technology were highlighted.

Chapter 3

Data Collection

Data collection in keystroke dynamics is not a difficult task, since no extra hardware is needed and keyboards are standard input devices for most computer systems. Logging keystrokes requires software to monitor key events on the keyboard and log the data. When collecting keystroke data, the user needs not to interact with the data logging software in any special way; they can just do their computer-based tasks as per usual. However, accurately collecting keystroke data is challenging as there are technical and ethical issues that have to be taken into consideration. If not properly addressed, these issues can tamper with the integrity of the collected data. Some these issues include the type of text, keyboard specifications and layout, and environmental conditions.

This chapter details the process of collecting data and how the technical and ethical issues surrounding the data collection were addressed. Section 3.2 provides a detailed description of how the final dataset was constructed. The collection of a free-text real-world dataset is one of the main contributions of this dissertation.

3.1 Collecting Data

There are various methodologies that can be followed when collecting keystroke data. The choice of the methodology may be influenced by the application in which KSD is being applied and the aims of the research. The following are the main questions that one needs to answer before starting with keystroke data collection:

- Artificial or real-world scenario?
- Free-text or fixed-text?
- What to record?

- Restricted or unrestricted collection?

The considerations that have to be made in order to answer these questions are discussed in the paragraphs that follow.

Ideally, data used for typist authentication would come from real-world recording where users are recorded as they perform their usual typing activities. However, due to security, trust and ethical issues it is not easy to convince a user to allow for their keystrokes to be recorded whilst they are busy with their typing activities. This is due to the fact that it is most likely that confidential data will be logged. The main benefit of monitoring real behaviour is that the user is unlikely to be affected by the limitation of an artificial task which is the unwillingness to continue partaking in the data collection experiment. It is common in studies where data is collected in an artificial setting for users to be unwilling as being part of the experiment may be a time consuming task whereas in a real-world scenario the user is in some way compelled to perform the typing tasks. This limitation results in a dataset consisting of too little data, as evident in Nisenson *et al.'s* dataset [32].

The question of free-text or fixed-text mainly depends on the application in which keystroke dynamics is being applied. In free-text data analysis, the sequence of keys and the length of keystrokes are not known as opposed to fixed-text analysis where a known key sequence with fixed length is expected. As alluded before, free-text analysis is typically suited where continuous authentication is the objective and where the user is typing in data into the system very frequently. On the other hand, fixed-text analysis is employed to make conventional authentication methods like PINs and passwords more robust at the initiation of a session.

When coming to the question of which parameters to log, issues of security and trust always arise especially for applications where there is sensitive personal data such as passwords and e-mails. With keystroke data logging unfortunately, the data entered into hidden password fields can be logged together with deleted data. Furthermore, the original text data can be regenerated by mapping the recorded keycodes to their corresponding characters and in some studies the raw text data is stored as well, an example of this is [27] where logging was performed in an e-mail client software. The aim of anonymous data collection is to ensure that the recorded parameters cannot be associated with the identity of the actual participant. Additionally, working with sensitive personal data required researchers to take strict data handling and processing measures to ensure the user's confidential data is well protected. In [27], the recorded e-mails were screened and confidential text data was deleted. Lastly, the choice of recorded parameters may limit

the usage of a dataset, this is seen in the case of GP's dataset [33] in which key release events were not recorded which implies that the hold-time feature cannot be calculated and thus may prevent parties interested in using the hold-time feature from using the dataset. In most studies that took place post GP's study, both key press and release events are recorded which addresses the limitation of GP's dataset [27, 34, 22, 35, 36].

Inputting data into the computer from the keyboard is required by various computer programs which include text editors, web browsers and games. This means that keystrokes can be monitored across a multitude of applications thus some studies exploit this by implementing unrestricted data loggers that record all keyboard events that are triggered in the operating system [35, 37, 38, 39]. Most studies however, restrict the monitoring of keystrokes to a specific platform for example within a text editor or web browser.

In this study, the keystroke data was collected in a real-world setting on free-text input and the collection was restricted within the text-area of the in-browser code editor. The next section discusses the process of collecting keystroke data in Moodle's Virtual Programming Lab (VPL) plugin.

3.1.1 Recording Keystroke Data on Moodle's VPL

In February of 2019, 283 second year engineering students at the University of Johannesburg who took a computer programming course offered by the Department of Electrical and Electronic Engineering Science gave consent to participate in a keystroke dynamics research study, thereby agreeing for their keystroke data to be logged whilst they performed given computer coding tasks on the course site. The course was published on Moodle, which is an open-source Course Management System (CMS) aimed at providing stakeholders in education with a single robust, secure and integrated system to create personalised learning environments [3]. The data collection was performed on two separate Moodle sites, namely the practice site and the assessment site. The practice site ran for a period of 15 weeks, from 20 February 2019 till 6 June 2019 and the assessment site ran for a period of 12 weeks, from 14 March 2019 till 6 June 2019. For a detailed schedule for the data collection particularly on the assessment site consult Appendix A.

The data collection happened in both controlled and uncontrolled environments where the controlled scenario was observed during the assessments and the uncontrolled setting was observed when participants attempted the practice questions on their own. During assessments, identical computers and keyboards were used to prevent any variations in the hardware and all the computers ran Windows. This was not difficult to achieve as

the assessments were performed in one of the university's computer labs and the teaching assistants were there to supervise and ensure that the set rules were adhered to by the participants. An assessment session ran for an hour and three sessions were conducted on each assessment day. Each student could only book a single session a week, therefore, each student was afforded at least 10 assessment opportunities in total. The opposite was true for when the participants did work on the practice site because they could access it any given place and time, so they could use their own desktops or laptops. Additionally, it was observed during tutorial sessions where the participants used the practice site, that students assisted each other and the teaching assistants also assisted here and there. This assistance was sometimes in the form of typing a line of code or two. This meant that some participants provided typing samples for online profiles that do not belong to them.

Since the course was about computer programming, an open-source plugin for Moodle, the VPL by [40] was employed to manage the programming assignments in Moodle, which includes the editing, running and evaluation of programs written by students. The VPL uses Ace, an open-source embeddable code editor written in JavaScript that runs in a browser and has syntax highlighting for more than 110 computer programming languages [41]. The data logging software was implemented to log the keystroke events triggered in the code editor's text field on a Hypertext Markup Language (HTML) form using client-side JavaScript (JS). When the `keydown` and `keyup` events were triggered, the times[1] of the events together with the keycode of the key were recorded; the text that was typed in by the participant was not recorded. The collected keystrokes were then saved to a `.csv`[2] file when the user hit the "save" button. All this took place in the background while a participant was attempting to solve a given computer coding challenge.

Every time when the participant hit the "save" button to save his/her code, the logger wrote all the recorded keystrokes to a user specific `.csv` file. The filename was created using the pre-fix "user_", followed by the user ID which is used to uniquely identify each user on the site and a timestamp to ensure uniqueness for each file thus resulting in a filename in this form "user_userid_timestamp". For example a file with the filename "user_1_1537707461.6787524.csv" would mean that the data file belongs to participant # 1, the file has a timestamp of 1537707461.6787524 and its file extension is `.csv`. The contents of the file are organised in a tabular format which consists of three columns with

[1]The time of the key events was obtained using the built-in JS function `Date.now()`. The function returns the number of milliseconds since January 1, 1970 00:00:00 UTC.

[2]`.csv`- Comma-Separated Values (CSV) is a file type for a plain text file that contains a list of data arranged in a tabular format where a comma or semi-colon is used as a delimiter.

the following headings: *press_time*, *release_time* and *key_code*, respectively. The contents of a raw keystroke data file are arranged in this form:

```
press_time,release_time,key_code
1537707376524,1537707376530,73
1537707376621,1537707376707,79
1537707376819,1537707376924,83
1537707377084,1537707377164,84
1537707377348,1537707377427,82
1537707377516,1537707377619,69
1537707377780,1537707377860,65
...
...
...
```

Each row in the example above represents one keystroke which is composed of a press-time, release-time and the keycode of the key that was pressed. The values are separated by commas. Each very large number is the timestamp, which is time in *ms* when a corresponding key was pressed or released. The number of rows in a data file is determined by the number of keys a participant presses before hitting the "save" button. In the above example, key 'I' (keycode = 73) was pressed at 1537707376524 and released at 1537707376530, the second key 'O' (keycode = 79) was pressed 1537707376621-1537707376530 *ms* later, and so on.

3.1.2 Technical Issues

Technical issues are quite a challenge, especially when collecting data in an uncontrolled setting. The remedies to most the issues that arise are careful programming and setting strict rules for the participants of the experiment. An additional remedy used to limit these issues in a controlled setup is, supervision, which is impossible to achieve for most real-world data collection scenarios. In this study a bit of both worlds was experienced, since data was collected on the assessment site which is only accessible in the university computer labs at a set time and on the practice site which is available to participants at any time.

The variations that were introduced by the uncontrolled setting gave rise to some of the technical issues discussed in the sections to follow.

3.1.2.1 Keyboard Layout

In this study, the computer keyboard of choice was the QWERTY[3] (US) keyboard layout as shown in Figure 3.1. The reason for choosing this layout was that it is the most used layout in South Africa and the world as a whole which made it the most viable layout compared to other existing QWERTY layouts and non-QWERTY layouts such as AZERTY and Dvorak respectively, to name but a few. The US and UK QWERTY variations have minor differences, for example, the UK one consists of 62 keys whereas the UK version has only 61 keys, on the typewriter keys [42]. Another noticeable difference is that, the `Enter` key on the UK version spans two rows and on the US version it only spans one row [42]. The effects of the mentioned differences on ones typing behaviour may seem uninfluential but in actual fact they play quite a role.

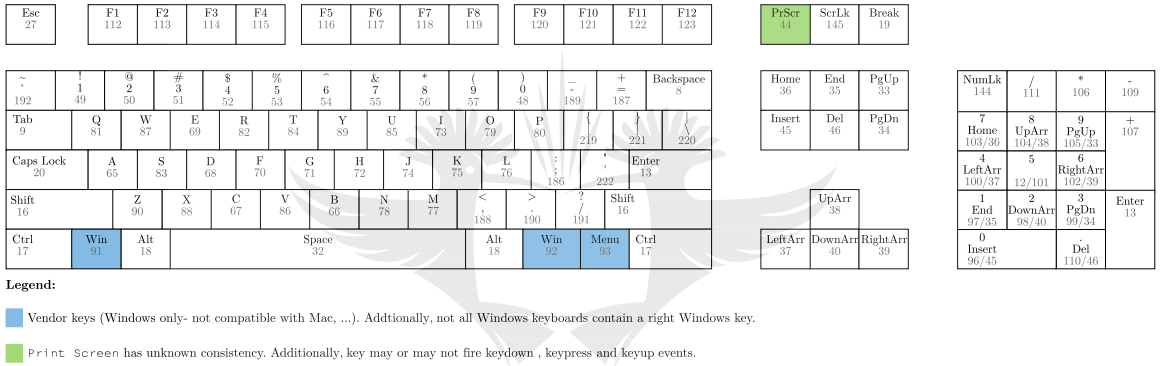


Figure 3.1: QWERTY keyboard layout with keycodes.

3.1.2.2 Browser

In order to determine the keycodes of each key on the keyboard the logging software was programmed to log the keycodes of the pressed keys on the console of the browser. Each key on the keyboard was then pressed and its corresponding keycode was recorded; then Figure 3.1 was created which is a mapping of each keyboard key and its corresponding keycode. To verify if the keycodes of all the keys were consistent across all major browsers, the same process of determine keycodes was followed but using different major browsers such as Chrome, Internet Explorer (IE), Firefox, Opera and Safari. After checking for consistency of keycodes across different browsers it was observed that the key-to-keycode

[3]The QWERTY computer keyboard layout has two variations, the United States (US) and the United Kingdom (UK).

mappings were consistent. Furthermore, the `Print Screen` key had an undefined press-time and a defined release-time, which was consistent on Chrome, Firefox, Safari and Opera whereas in IE the key did not fire at all. The cause of this may be that the `keydown` event of this key has been suppressed on the browsers that were tested. In the case of using the IE, both the `keydown` and `keyup` events did not fire. Lastly, to ensure that accurate keycodes are obtained, measures were taken when programming the data logger to have it to work consistently across the mentioned major browsers.

3.1.2.3 Ace Code Editor

When using the Ace code editor, the `keydown` events of the following keys: `Backspace`, `Tab`, `Page Up`, `Page Down`, `End`, `Home`, `Left Arrow`, `Up Arrow`, `Right Arrow`, `Down Arrow`, `Print Screen`, `Insert` and `Delete` did not trigger, however the `keyup` events did trigger. This behaviour resulted in all the mentioned keys to have an undefined press-time and a defined release-time. This challenge was due to how the `keydown` events for the keys were handled by the code editor; instead of using the default browser event handlers the developers implemented their own event handlers for the listed keys. To overcome this challenge edits were made in the default *ace.js* file which contained the main JS code for running the editor. When examining the code it was found that the default event handling was suppressed by making the event handler to return `false` and/or by calling `event.preventDefault()` and `event.stopPropagation()`. To overcome this challenge, the `keydown` event listener of the keystroke logger was implemented in the *ace.js* file before the code that prevented the events from triggering.

3.1.3 Ethical Issues

Since the study involved the collection and use of personal behavioural data, measures were taken to ensure that the data collection process is executed ethically. The first step in this quest was to apply and obtain ethical clearance from the University of Johannesburg where the study was conducted. An informed consent form that provided the purpose of the study, a detailed description the data collected and measures that would be taken to protect participants and their personal data, was created on the course site and made accessible for every student who took the programming course. The data that was used in this study was that of the students who gave consent.

It is of paramount importance that all the ethical issues are addressed before the dataset is utilized. Furthermore, the dataset cannot be shared with any party attempting to replicate the study if it contains sensitive personal information. The e-mail address provided

by the participant during registration was solely used for the purpose of identification when accessing the site and it shall not be shared or released publicly to ensure confidentiality and anonymity. To identify the various participants within the study, each participant was given a unique ID when they registered on the site. This ID is not in any way linked to the personal information (e-mail address) provided by the participant. Lastly, the data logger was limited only to the code editor, this was done to prevent logging of the participants' sensitive data such as their emails and passwords. This makes sense since the focus of this study is not on static authentication or password hardening but rather on dynamic keystroke analysis.

Most participants were hesitant to partake in the study even though assurance was given that the use of their keystroke data in the study would not affect their academics in any way, thus some students chose not to participate and the others opted to stop participating in the study after some time. The main reason for this was that some students felt and thought that the usage of their typing data and the experimental findings may be used to implicate and defame them.

Lastly, the results obtained and the observations made in this study will not be used to implicate the participants. There will not be any form of discrimination of participants, all the collected from all the participants shall be used. All efforts shall be made to ensure that the participants are protected from any potential harm with regards to their reputation and that of their institution.

3.2 Final Dataset

Table 3.1: Numerical summary of our datasets.

Dataset	Users	No. Keystrokes	Avg. Keystrokes/User
Practice	283	6M	21,000
Assessment	270	3M	11,000

Table 3.1 gives a numerical summary of the outcomes of the data collection process. Of the 323 potential participants, 283 consented, 31 declined and 9 abstained. The data samples of the 31 students who declined to be part of the study were removed from the final dataset. Of the 283 participants who consented, 13 did not sit for assessments thereby leaving 270 participants who provided typing samples on the test site. From the practice site, just above six million keystrokes were recorded from 283 participants

over the data collection period with a each participant providing approximately 21,000 keystrokes on average and the most prolific participant providing 129,436 keystrokes. From the assessment site, just above three million keystrokes were recorded with an average of 300,000 keystrokes per session and each participant providing approximately 11,000 keystrokes on average.

3.2.1 Dataset Description

The complete dataset is packaged in two `.csv` files, one containing keystroke data collected from the practice site and the other with data collected from the test site. The file structure of both files is the same and is arranged as follows:

- The data is arranged in tabular form with 4 columns.
- The first column, *user_id*, is an unique identifier for each participant, ranging from 1 to 283.
- The second and third columns store the timestamps at which press and release events were triggered, respectively.
- The last column is for the keycode.

Consider the following example of how the contents of the data files are arranged:

```
user_id,press_time,release_time,key_code
3,1537707376524,1537707376530,37
3,1537707376621,1537707376707,13
3,1537707376819,1537707376924,83
...
...
...
```

The example above presents keystroke data from participant #3. The first key which this participant typed was the `Left-Arrow` key (keycode = 37) and the key was pressed at time 1537707376524 and released 6ms later at time 1537707376524. The second key that the user typed is the `Enter` key which was pressed at time 1537707376621, 91ms after the `Left-Arrow` was released, and so on.

3.3 Conclusion

This chapter presented the data collection methodology followed in this dissertation. The chapter also touched on various technical and ethical issues encountered during the data

collection phase and how they were addressed. The chapter was concluded by presenting the description of the final dataset.

Chapter 4

Dataset Analysis

In this chapter attention will be drawn to the analysis of the dataset that has been constructed as explained in the previous chapter. The analysis includes the pre-processing of the raw keystroke data such as performing outlier removal and various data transformations. The extraction of timing information used to create the features will be discussed together with other non-timing features that are going to be used. An analysis of the features will be performed whereby an investigation will be carried out to determine the most important features for this application, this will be done in Section 4.2.4. Additionally, reasons and motivation will be given as to why the chosen features were selected over the other available ones.

4.1 Raw Data Pre-Processing

Pre-processing was performed on the raw keystroke data in order to prepare the data for feature extraction. Two operations were performed on the data, which are: the removal of erroneous data points and sorting. The data logger software module was implemented to return a value of -1 for keys that have undefined press or release times; thus data rows that have a value of -1 for the press or release time were removed from the dataset.

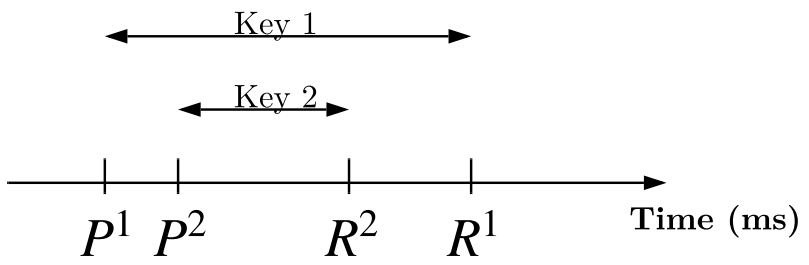


Figure 4.1: Overlapping key sequence of two keys.

Consider the scenario shown in Figure 4.1 and assume that "Key 1" is a modifier key and "Key 2" is one of the typewriter keys on the keyboard. If a typist wants to type the character '#', they would typically follow the key sequence shown in Figure 4.1 and "Key 1" would correspond to `Shift` (keycode = 16) and "Key 2" to '3' (keycode = 51). From observing the sequence as depicted in the figure it is evident that P_1 is lesser than P_2, however a keystroke can only be logged successfully after the `keyup` event of that particular key has been fired. For this reason, in the data file the data row of '3' would appear before that of `Shift` even though `Shift` was pressed first. This called for a second operation on the raw data. This operation involved sorting data rows in ascending order using the key press-time. Performing the sort is important as it preserves the original sequence of the keys; not performing the sort would result in incorrect calculations of the inter-key time features.

4.2 Features

In this section, the information extracted from the data will be discussed. This section's pre-requisite is Section 2.2 which gives a theoretical background on the timing features and the mathematics involved in extracting them. The equations used to calculate the features have been restated below, for the sake of easy referencing.

$$t_h^i = R^i - P^i \tag{2.1}$$

$$t_{ud}^i = P^{i+1} - R^i \tag{2.2}$$

$$t_{dd}^i = P^{i+1} - P^i \tag{2.3}$$

$$t_{uu}^i = R^{i+1} - R^i \tag{2.4}$$

$$t_{du}^i = R^{i+1} - P^i \tag{2.5}$$

4.2.1 Extraction

Feature extraction is the process of making meaning of the collected data– the generation of information from raw data. Using the parameters recorded during data collection, i.e., press-time, release-time and keycode, the KSD features were extracted. The timing features were computed by applying (2.1) through (2.5) to the press and release timestamps. The keycode values were taken as they are from the data file, no calculations were made to extract them. The extracted features were then stored to construct feature vectors as shown below where V is a vector and the subscript is the feature contained by the vector:

1. The generated V_{t_h} feature vector is as follows:

$$V_{t_h} = \{t_h^0, t_h^1, t_h^2, ..., t_h^n\} \tag{4.1}$$

where the position of a particular key event or character in the input stream is denoted by n. The size of the feature vector is $n + 1$.

2. The generated $V_{t_{ud}}$ feature vector is as follows:

$$V_{t_{ud}} = \{t_{ud}^0, t_{ud}^1, t_{ud}^2, ..., t_{ud}^n\} \tag{4.2}$$

3. The generated $V_{t_{dd}}$ feature vector is as follows:

$$V_{t_{dd}} = \{t_{dd}^0, t_{dd}^1, t_{dd}^2, ..., t_{dd}^n\} \tag{4.3}$$

4. The generated $V_{t_{uu}}$ feature vector is as follows:

$$V_{t_{uu}} = \{t_{uu}^0, t_{uu}^1, t_{uu}^2, ..., t_{uu}^n\} \tag{4.4}$$

5. The generated $V_{t_{du}}$ feature vector is as follows:

$$V_{t_{du}} = \{t_{du}^0, t_{du}^1, t_{du}^2, ..., t_{du}^n\} \tag{4.5}$$

6. The generated V_{fk} feature vector is as follows:

$$V_{fk} = \{fk^0, fk^1, fk^2, ..., fk^n\} \tag{4.6}$$

7. The generated V_{tk} feature vector is as follows:

$$V_{tk} = \{tk^0, tk^1, tk^2, ..., tk^n\} \tag{4.7}$$

The size of the inter-key time vectors, 4.2 through 4.5 is one less than the size of V_{t_h} due to the fact that the first key in the typing sequence would not have inter-key times; therefore the inter-key time values of the first key in the input stream vectors should be removed, this is to ensure that all the feature vectors are of the same size. Additionally, for the first key in the input stream, $tk^0 = fk^0$.

4.2.2 Outlier Detection and Treatment

When a person is typing, they do not type at constant speeds throughout and they are most likely to take pauses during the act. These pauses maybe attributed to thinking time and other interruptions. These pauses translate to high valued inter-key times. Another scenario is when a person types in a word they are used to typing, here muscle

memory comes into play and the person types in the word with ease which translates to low valued hold times and inter-key times. These timing values that are either too large or too small are outliers and they do not necessarily reflect the user's actual typing behaviour. An outlier detection and treatment process was employed to handle such data points.

There are numerous outlier detection taxonomies, the most basic being the extreme-value analysis[1]. This group of outlier detection methods is applicable to 1-dimensional data and the key is determining minimum and maximum bounds to separate inlying points from the outlying ones as shown in Figures 4.2 and 4.3. In the subsequent sub-subsections the outlier detection methods explored in this dissertation are discussed together with some of their benefits and drawbacks.

4.2.2.1 Standard Deviation Method

This method[2] can be employed given that the distribution of the values in the sample is Gaussian or Gaussian-like, we can use statistical measures, namely, the standard deviation and the mean of the sample to compute the bounds used to identify outliers. This method exploits the property of the Guassian distribution that the standard deviation from the mean can be used to summarize the percentage of the values in the sample. An outlier is then a data point x_m that lies outside the bounds, i.e.:

$$x_m < \mu - k\sigma \cup x_m > \mu + k\sigma \tag{4.8}$$

where:

μ is the mean of the sample.

σ is the standard deviation of the sample.

k is a constant that denotes the number of standard deviations from the mean.

[1]Refer to [43] for a comprehensive discussion of outlier detection methods and theory relating to outliers and their analysis.

[2]An equivalent of this method is using the standardised Gaussian which has a mean of zero and standard deviation of unity ($\mu = 0, \sigma = 1$). This version uses the z-score to compute the bounds.

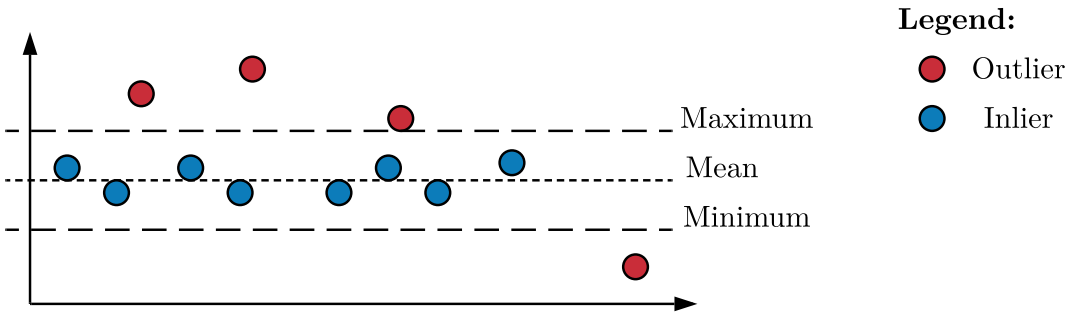


Figure 4.2: Mean and standard deviation outlier detection method.

When using this method the common practice is selecting a k value of 2 or 3 which preserves 95.5% and 99.7% of the values, respectively. What makes this method attractive is that it is easy to understand and implement, however, it suffers from one main drawback. The drawback is that the outlying values are also included in the calculation of the bounds which means that these extreme values influence the position of the bounds therefore making the method less robust on outliers. Moreover, this method assumes that the timing data follows a Gaussian distribution which may not always be true as timing data is more likely to follow a Rayleigh distribution.

4.2.2.2 Interquartile Range Method

Not all data is Gaussian or Gaussian-like, a good statistic for summarising non-Gaussian data is the Interquartile Range (IQR). The IQR is calculated by computing the difference between the 75^{th} and the 25^{th} percentiles of the data. Using the IQR, we can define bounds on the sample values that are a factor k of the IQR below and above the 25^{th} and the 75^{th} percentiles, respectively. An outlier is then a data point x_m that lies outside the bounds, i.e.:

$$x_m < Q1 - k \times IQR \cup x_m > Q3 + k \times IQR \tag{4.9}$$

where:

$IQR = Q3 - Q1$

k is the IQR multiplier.

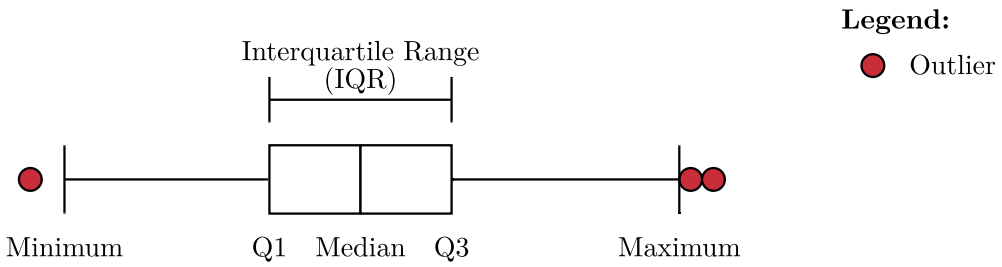


Figure 4.3: IQR outlier detection method.

Compared to the standard deviation method, the IQR outlier detection method is more robust to outliers as it uses statistical measures that do not include extreme values to calculate bounds.

Outlier detection is followed by treatment– what to do with the outliers. Outliers can be treated in one of two ways, namely, elimination or replacement. When using elimination as a remedy, the detected outliers are removed from the dataset and in the case of replacement, the outlying data points are replaced with a certain value, usually the mean or median score. In applications where there is limited data points the most viable outlier treatment would be replacement, however, where there is abundant data points removal is usually the solution.

Based on the discussions presented in this section, for this dissertation, the IQR based technique was used for outlier detection due to its robustness on outliers. With regard to outlier treatment, all the detected outliers were removed from the dataset reason being that replacing outliers with the mean, median or any other value would be artificially creating the user's typing behaviour which may distort the user's actual behaviour.

4.2.3 Scaling

Feature scaling[3] is a technique used to standardise the range of independent variables or features of data. Feature scaling is a very important step during feature pre-processing especially when the features are of varying magnitudes. This pre-processing step may

[3]Feature scaling is relevant to continuous variables therefore it was only applied on the timing data.

vary the results when using distance-based algorithms or algorithms that assume normality and have a little or no effect in tree-based algorithms as they can handle features of varying magnitudes. Due to the algorithms employed in this dissertation, feature scaling is necessary because hold times are typically less than a second whereas inter-key times are usually a couple of seconds therefore feature scaling is necessary so that all features are on the same level, magnitude-wise.

There are four commonly used scaling methods, namely, standardisation, mean normalisation, min-max scaling and scaling to unit length. The standardisation scaling technique is presented here as it was was employed in this study[4]. Feature standardisation makes values of each feature to have a mean of zero and unit standard deviation. This is achieved by replacing each sample point, x_m, within a particular feature vector by its z-score, z_m, which can be calculated as

$$z_m = \frac{x_m - \mu}{\sigma} \tag{4.10}$$

where:

μ is the mean of the feature vector.

σ is the standard deviation of the feature vector.

4.2.4 Dimension Reduction

Dimension reduction is a technique used to reduce the dimension of used features (the number of features used). This technique transforms a set of features, $\mathcal{V}_{original}$, to a new set of features, $\mathcal{V}_{new}$, where $\mathcal{V}_{new} < \mathcal{V}_{original}$ [45]. This can be achieved by manually examining the features and selecting only the most informative ones. An alternative is employing algorithms that take in the original features as input to produce new features with more or similar informative power. Some of the algorithms that may be employed for this task include the Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA) and Autoencoders (AE), to mention but a few. One main drawback with using this approach of dimension reduction is that unlike the original features, the newly generated features may not be interpretable by humans. However, to the computer these new features will have more or less the same ability to train learning agents. Some of the advantages of dimension reduction include [45, 46]:

- Elimination of highly correlated and redundant features.
- Less computing resources are used as space and memory required to store data is reduced due to the decrease in the dimensionality.

[4]Refer to [44] for a discussion of the other mentioned scaling techniques.

- Reduced training times for learning agents.
- Dimension reduction to two or three dimensional spaces makes it easier to visualise the data as humans can only interpret plots up to three dimensions.

PCA[5] was applied on the timing features to ensure that any heavily correlated features are eliminated as they introduce unneeded redundancies. Figure 4.4 shows the information contained in the PCA generated features. From the plot it can be observed that the 4^{th} and 5^{th} principal components contain almost no information therefore the dimension of the timing features was reduced from $5D$ to $3D$ while still preserving almost all the information.

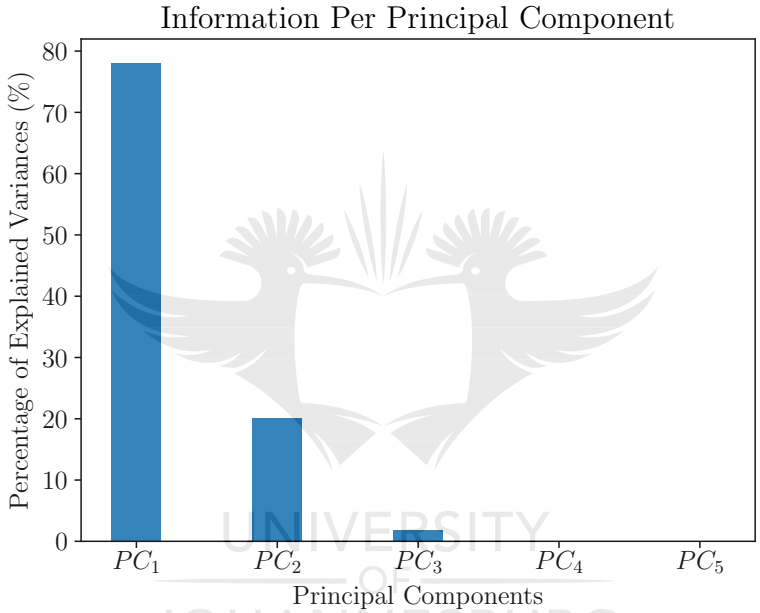


Figure 4.4: Explained variance per principal component which shows the amount of information stored in the principal components.

Instead of using the original timing features, the PCA generated features were used. The features that contributed the most in terms of information content are: t_h, t_{ud} and t_{dd} (in order of importance). The other two inter-key timing features have negligible information content because they are they are well correlated with the other timing features. To further demonstrate the need of dimension reduction, linear combinations of equations 2.1 through 2.5 were taken in order to determine the equation(s) that are correlated.

[5]This algorithm assumes data normality therefore the input data was scaled as explained in Section 4.2.3.

From this investigation, the following observations were made:

$$(2.1) + (2.2) = (2.3)$$
$$R^i - P^i + P^{i+1} - R^i = P^{i+1} - P^i$$

$$(2.1) + (2.4) = (2.5)$$
$$R^i - P^i + R^{i+1} - R^i = R^{i+1} - P^i$$

$$(2.3) + (2.4) = (2.2) + (2.5)$$
$$P^{i+1} - P^i + R^{i+1} - R^i = P^{i+1} - R^i + R^{i+1} - P^i$$

The linear combinations listed above corroborate the results achieved using the PCA regarding the most useful/informative features.

4.3 Conclusion

In this chapter, the data processing procedures employed were presented which included the raw data pre-processing, feature extraction and outlier detection and treatment where various methods were explored and the optimal one selected. Furthermore, dimension reduction by means of the PCA was employed to remove redundant features.

Chapter 5

Detection and Elimination of Common Typist Traits using OC-SVM

In this chapter[1], we propose a method of detecting and eliminating typing traits that are common amongst users in a dataset using a machine learning based novelty detection algorithm.

The remainder of this chapter is arranged as follows. The immediate section to follow provides an introduction and discusses the motivation behind the work. In Section 5.2.1, we present the theory of one-class classification and the One-Class Support Vector Machines (OC-SVM) algorithm which was used to devise the proposed approach. In Section 5.3, we discuss the operation of the suggested approach. In Section 5.4, we present the experimental method followed to test the suggested approach together with the evaluation metric used, the results obtained and their discussion. This chapter will then end of with remarks– the strengths and shortcomings of the proposed approach.

5.1 Introduction and Background

When analysing the typing data that was used in our study, it was noted that there are common digraphs that most users type. For example, the top ten most typed digraphs are (in descending order): `Backspace Backspace`, `Right-Arrow Right-Arrow`, `Left-Arrow Left-Arrow`, `Shift 9` (to produce '`(`'), `in`, `Down-Arrow Down-Arrow`, `nt`, `Up-Arrow Up-Arrow`, `; Enter` and `t Space`. It is worth noting that the `Backspace Backspace` digraph was typed approximately three times more than the

[1]The work presented in this chapter was published in [47].

second most typed digraph. Majority of the most typed digraphs are unprintable characters and they would typically be attributed to the editing of free-text.

Combined, the top three most typed digraphs accounted for 15% of all the data that was collected. Given that the keystrokes were collected while users were typing computer code it makes sense that the keystrokes that dominate are those associated with editing of text because when coding one usually has to debug and make corrections to their code. Additionally, the participants are novice computer programmers and majority are unskilled typists, therefore making them prone to typing errors compared to someone who is a seasoned programmer and/or typist. Of importance is that the unprintable characters may not necessarily reflect the users typing behaviour as they occur as a result of the user holding down the key for extended periods of time. The KSD that dominate in this situation are actually the repetition rate of the operating system, not the typing behaviour of the user.

According to information theory, observing an event that has a low probability of occurring is more informative compared to observing an event that has a high probability of occurring, this concept is referred to as information entropy [48]. Therefore, using these common digraph as features would be ineffective as they are not good discriminating features for a typist since they occur amongst a lot of subjects thus making them less informative. To improve classification performance, these common features should be ignored/removed as they carry little to no information.

In Information Retrieval (IR) particularly the sub-field of text retrieval, words that occur in all or most documents in the collection are usually discarded as they are regarded to have little or no discriminatory power. This is achieved by developing a list of all words that are used excessively in a language [49]. For example, stop words in the English language include “is”, “in”, “the” and “a”, to mention but a few. In text retrieval literature these words are known as stop words and a list of stop words is referred to as a stop list. When performing text processing in text retrieval systems such as search engines and document classifiers, any words that appears in the stop list are ignored when querying results or performing classification. By so doing, the focus is given to the keywords that are presented in the provided text. An additional benefit of the removal of stop words is that it helps save storage space and processing time.

Currently in KSD literature, not a lot of attention has been given to investigating how commonly typed n-graphs affect the performance of keystroke analysis algorithms. Most

algorithms proposed in literature work on the idea that people are unlikely to perform the same task in the exact same way. Therefore, they use common typing features (such as digraphs) that occur amongst users and base their decision on how uniquely each user types the digraphs [27, 18, 33]. It is worth mentioning that the authors of [18] and [27] acknowledge that not all digraphs are effective when discriminating users. From investigation, [18] found the digraphs: `in`, `io`, `no` and `on` plus one other digraph that could be `ul`, `il` or `ly` could be used to perfectly authenticate typists[2]. The findings made by [27] regarding the importance of digraphs with relation to their discriminatory power corroborates the findings of [18].

Unlike [27, 18, 33] where common/shared digraphs are used to discriminate between users, we follow a different approach, one similar to that used in text retrieval regarding frequently occurring features, i.e. discarding what is very common and using what is unique to discriminate between users. Even though we agree with the findings of [27] that common typing behaviour amongst users is typically associated with how most users interact with keys associated unprintable characters, we hypothesise that the common behaviour may not be limited to digraphs that contain unprintable characters only but may also include other digraphs. Therefore, we present a OC-SVM based method to remove digraphs that are common between users. This is done with the aim of improving the discriminating ability of a classifier while reducing the amount of data that needs to be processed.

5.2 One-Class Classification

One-Class Classification (OCC) is a classification approach where a learning agent is trained to identify objects of one specific class, i.e. target class, by learning from a training set that only contains examples of the target class. The learning agent then learns a decision boundary that encloses the data that belongs to the target class within the boundary. When used for prediction on previously unseen data, the agent can make two possible judgements- *target* if the observed data lies within the set decision boundary, or *unknown* if the observed data lies outside of the boundary, this is illustrated in Figure 5.1.

OCC learning algorithms are mainly used for novelty detection, outlier detection and anomaly detection. Therefore one-classification is also referred to as novelty detection or outlier detection, depending on the context and application. One-class learning is usually

[2]In the experiment conducted by [18] six professional secretaries were used and the classifications were made by hand.

useful in situations where we have little or no negative examples but positive examples are in abundance and we want to train a learning agent that would distinguish positive examples from the negative ones. In the case of identity verification using KSD, one-class classification is useful as we are only guaranteed of data from the legitimate user as it is infeasible and impossible to collect typing samples of all possible impostors. Some of the most used outlier/novelty detection algorithms in practice include one-class Gaussian, one-class K-means, one-class kNN, and one-class SVM.

In the following section, we briefly look at the theory of a novelty detection algorithm, the OC-SVM, which is a key component of the suggested approach used to detect and eliminate common typing traits.

5.2.1 One-Class Support Vector Machines

OC-SVM are a specialised case of Support Vector Machines (SVM) by [50] intended for unsupervised one-class learning. This algorithm employs a kernel function to perform a transformation on the input data, therefore, mapping the data into a high-dimensional feature space and finds the maximal margin hyperplane (decision boundary) that separates the training data from the origin. To perform this separation, the authors suggest solving a quadratic problem subject to some linear constraints[3]. Of the parameters in the quadratics program, the one most valuable to our application is ν, $\nu \in (0, 1)$, which is known as the contamination factor. This contamination factor is defined as the upper bound on the fraction of training errors and a lower bound of the fraction of support vectors. Simply put, the contamination factor corresponds to the probability of finding a new, but regular, observation outside the learnt decision boundary.

When used for predictions on a newly observed data point, x, this algorithm outputs a function, f, for each observed data point, i.e.,

$$f(x) = \begin{cases} +1, & \text{if } x \text{ lies within the decision boundary} \\ -1, & \text{if } x \text{ lies outside the decision boundary} \end{cases} \tag{5.1}$$

In the section to follow, we draw our attention to an important component of the OC-SVM, the kernel function. We provide a brief review of some of the available kernel options for the OC-SVM and discuss some of their strengths and weaknesses.

[3]To learn more about the quadratic program and its constraints, the interested reader may refer to [50].

5.2.1.1 Kernel Function

There are various kernel functions that can be employed to perform the transformation task, some of them include those listed in Table 5.1 which are the commonly used kernels for the OC-SVM. The decision boundary learnt by the OC-SVM is determined by the kernel function used, examples of the decision boundaries associated with the kernels in Table 5.1 are shown in Figure 5.1. The performance of the of the OC-SVM mainly depends on how well the learnt decision boundary separates the target class from the unknown class. Therefore, when selecting a kernel, one has to make a choice that will result in the model making the least number of errors.

Each of the kernel functions listed in Table 5.1 has pros, cons and typical scenarios in which it works best. The notable advantage of kernel functions: linear, polynomial and sigmoid are is that computations are usually fast compared to those associated with the Radial-Basis Function (RBF) kernel. These kernels work well in scenarios where the data is linearly separable, i.e. where there exists a line that accurately separates the target from the unknown class [51]. In our case, it is unknown whether the data we are working with is linearly separable or not. Therefore, a kernel that places data points to the left or the right of a linear separator may not be fit for the task.

Unlike the kernels discussed thus far, the RBF kernel creates a closed region around the data that is most likely to belong to the target class and all points that lie outside of that region are regarded as outlier. Due to its non-linear nature, this kernel works in scenarios where the data is linearly separable and where it is not (which is usually the case when with most datasets). Therefore, the performance of this kernel does not hinge on separability of the data which is quite attractive. From Figure 5.1, one can easily observe that the RBF out performs the other three kernels by a significant margin. Furthermore, the authors of [51] report that when working with the OC-SVM one should just use the RBF kernel and not bother trying other kernel functions as they are not fit for solving OCC problems.

Table 5.1: Widely used kernel functions [51].

Kernel Name	**Kernel Expression**
Linear	$\kappa(x_1, x_2) = \langle x_1, x_2 \rangle$
Polynomial	$\kappa(x_1, x_2) = (\langle x_1, x_2 \rangle + 1)^d$, with $d \in \mathbb{N}^*$
Sigmoid	$\kappa(x_1, x_2) = \tanh(\theta \langle x_1, x_2 \rangle)$, with $\theta \in \mathbb{R}$
RBF	$\kappa(x_1, x_2) = \exp(-\frac{\|x_1 - x_2\|^2}{2\sigma^2})$, with $\sigma \geq 0$

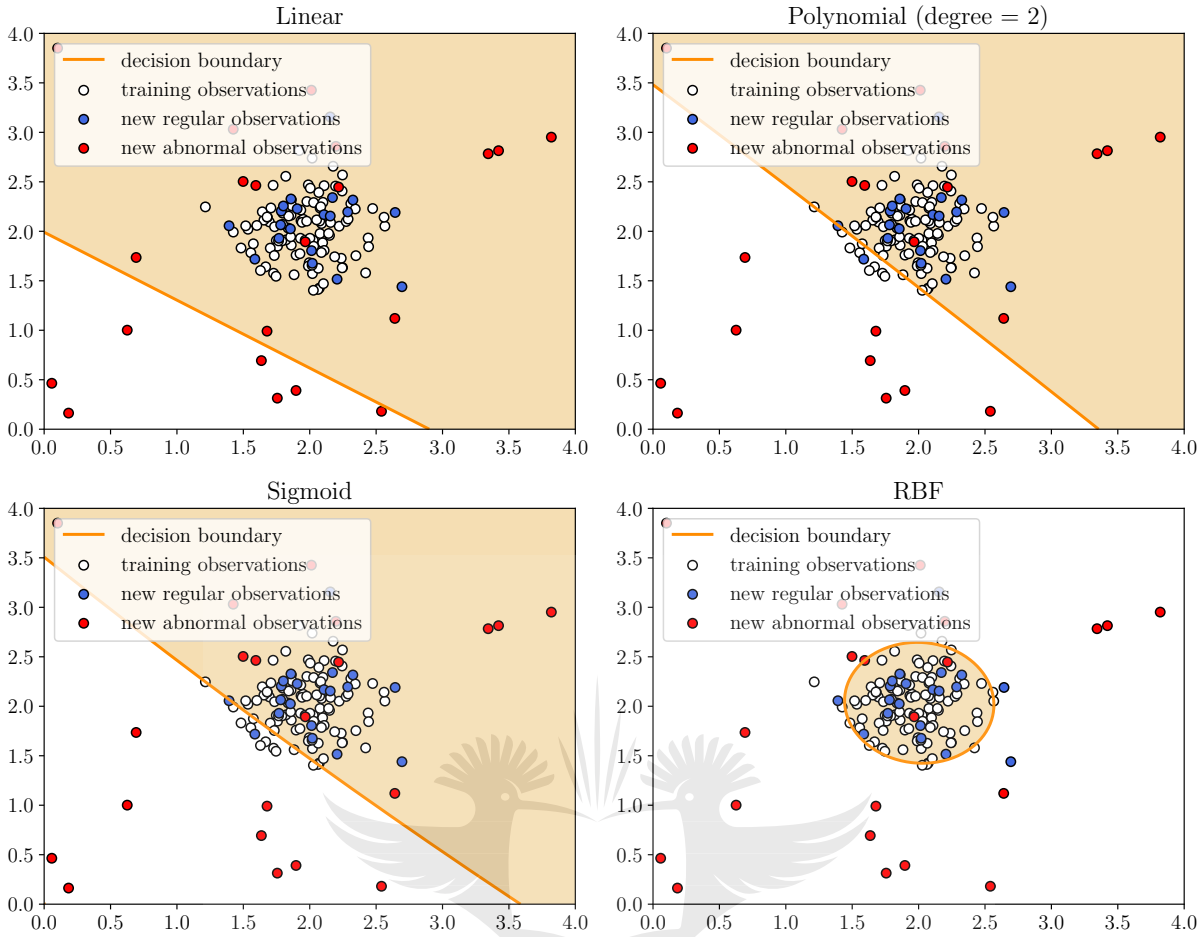


Figure 5.1: Examples of One-Class Support Vector Machines boundaries for the different kernel functions. Note: These examples were obtained on synthetic data as they are meant for illustrative purposes.

5.3 Approach

Identifying common typing behaviour amongst users in a dataset can be achieved by comparing the typing data of each and every user in the dataset while taking note of the keystroke data that is similar between the users. Our hypothesis is that there exists some shared typing behaviour between all or most users which can be identified by looking at the speed it takes for users to type certain digraphs. We believe that majority of these similar digraphs may include the aforementioned commonly typed digraphs, however, they are not limited to them.

Consider the Venn diagram shown in Figure 5.2 where each circle represents a user's typing behaviour. The shared typing behaviour between users can be defined as the intersection between the three circles in the Venn diagram. Therefore, given a typist

dataset with data from N typists, the common typing patterns that occur amongst all users, U_{super} (henceforth referred to as the *super user*), occur in the region given by:

$$U_{super} = U_1 \cap U_2 \cap U_3 \cap ... \cap U_N \tag{5.2}$$

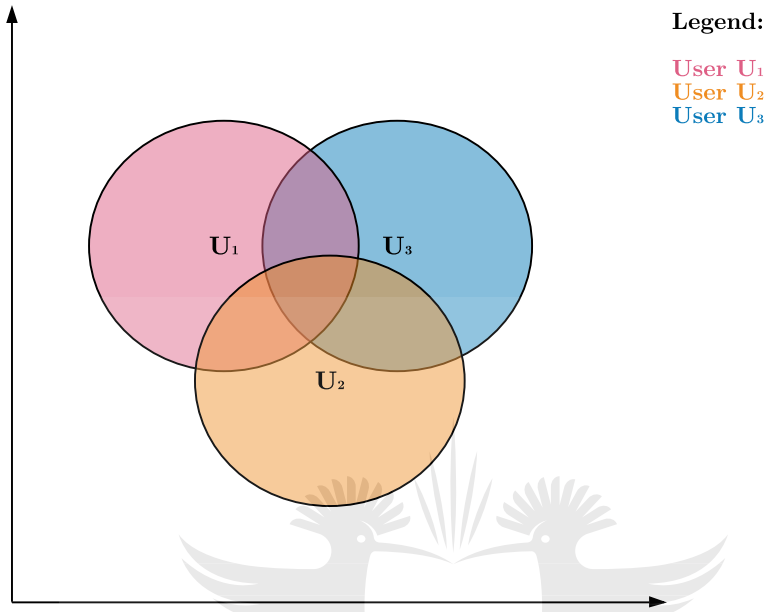


Figure 5.2: Venn diagram illustrating how user typing traits may overlap. Note: A circle represents a single user's typing behaviour/data.

The data points contained in the region described by 5.2 have little or no discriminating power as they are common. Given only that data, it is not possible to determine who it came from. Therefore, these common data points should be ignored during classification or removed from the dataset entirely. The distinctive typing traits occur in regions where a particular user's circle does not intersect with another circle- these are the regions we are interested in when classifying typists.

We exploit the characteristics of the OC-SVM with a RBF kernel to devise a method of finding and eliminating common typing traits amongst users which is as follows:

1. Due to the number of keystrokes, it is infeasible to use all the keystrokes in the dataset. Therefore, we randomly sample $500,000$ keystrokes from all the users in the dataset to create the *super user's* typing profile.

2. Perform feature extraction and pre-processing on the sampled keystrokes as described in Chapter 4. An additional pre-processing step was employed to convert

the keycodes from their number representation to categorical representation using one-hot encoding. Since there are 99 unique keys, this encoding scheme converted each keycode into a 1-dimensional array of zeros with a one at the index that corresponds with the integer label of the key as assigned by the encoder.

3. Instantiate the One-Class Support Vector Machines (OC-SVM)[4] with the following fixed hyper-parameter settings[5]:

   - Kernel: RBF. Based on literature presented in Section 5.2.1.1, it was almost obvious that the RBF kernel would out perform the other kernels for the OCC task at hand. However, to ensure that the best kernel choice was made, a grid search was performed to find parameter and hyper-parameter settings best suited for our OCC task. As expected, the most satisfactory results were obtained using the non-linear RBF kernel. Furthermore, the observations we made from performing the grid search corroborate the findings made by [51] that the RBF kernel typically gives the best results in OCC problems, particularly when using the OC-SVM.

4. Fit the OC-SVM model using the super user's typing data at varying ν values with $\nu \in [0.2, 0.8]$.

5. After every fit operation, the model is then applied on the data of each user in the dataset to determine the labels of each sample, $+1$ for inliers and -1 for outliers.

6. All the data samples that with a $+1$ were removed to create a new dataset that corresponds with a particular value of ν. This process is repeated for $\nu \in [0.2, 0.8]$ in increments of 0.1. The percentage of data removed is approximately $(1-\nu)\times 100\%$.

5.4 Experiment

To test and evaluate the proposed common data detection method, an experimental procedure was designed and followed. The experimental data was sourced from the top 50 keystroke contributors[6]. The test procedure and evaluation criterion are thoroughly detailed in the sections that follow.

[4] A public implementation of this algorithm by Sci-kit learn was used, see [52] and https://scikit-learn.org.

[5] All the other hyper-parameters were set to their default values.

[6] A subset of the users was used because as the number of users in the experiment increases, the number of keystrokes samples to create the super user's profile also has to be increased. The sample size of the data used to create the super user is proportional to the time taken to train the OC-SVM, therefore, its increase results in longer training times for the OC-SVM.

5.4.1 Methodology

In order to observe the effects of the data removal, a one-class classifier, namely, Gaussian Mixture Model (GMM)[7] was implemented. The classifier model was trained and tested using the reduced datasets obtained by varying ν. The GMM had the following hyper-parameter settings:

- Number of mixture components: 90
- Covariance type: Full
- Initialisation method: K-Means method for initialising initial parameters such as the weights, means and precisions.

These parameters were determined using a grid-search algorithm to optimise the hyper-parameters. Default values were used for the other GMM parameters which are not mentioned above.

To evaluate the suggested approach, an $\boldsymbol{N} \times \boldsymbol{N}$ square matrix, $\boldsymbol{S}$, was constructed by training a GMM model on single user's data and test it using the data from the N selected users. The rows of this matrix represented the actual user whose data was used to train the model and the columns represented the classified users (i.e. users whose data was used to test the model). The matrix was then populated using log-probability scores returned by the GMM. This procedure was performed at varying data removal percentage values.

To measure the effect the removal of common data has on the classifier's discrimination ability, the discrimination index (δ) was used as a metric. The discrimination index is a measure of how well a one-class classifier can discriminate between the normal and anomalous data. To compute δ, we iterate through the columns (test sets) of $\boldsymbol{S}$. In each column we extract the log-probability score that lies on the main diagonal, $\boldsymbol{S_{j,j}}$ and then determine the largest score in the column, $max(\boldsymbol{S_j})$, excluding $\boldsymbol{S_{j,j}}$. Using the $\boldsymbol{S_{j,j}}$ and maximum remaining value in the column, the per-column discrimination index, δ_j is given by:

$$\delta_j = \frac{\boldsymbol{S_{j,j}}}{max(\boldsymbol{S_j})} \quad \forall j \in [1, N] \tag{5.3}$$

To obtain the overall δ for the system, the mean of all the per-column indexes was computed. The discrimination index, δ, is always positive and larger values are better.

[7] A public implementation of this algorithm by Sci-kit learn was used, see [52] and https://scikit-learn.org.

5.4.2 Results

The results of this experiment are shown in Table 5.2 and Figure 5.3. The results show how the classifier discriminates between user as the datasets decreases in size due to the removal of common data. At 90% data removal, it was observed that there was insufficient data to train the models thus a $\nu \in [0.2, 0.8]$ was used.

Looking at the results, it can be observed that the classifier's performance improves as common data is removed in the interval $0\% - 40\%$. The improvement is shown the by the increase in both δ and δ_{min}, with the highest δ and δ_{min} both recorded at 40% data removal. This implies that around 40% of the data is common across all users and only 60% of the typing data is discriminating. The classifier's performance drops beyond the 40% removal mark, in the range $(40\%, 70\%]$. This drop is most likely due to the loss of data with sufficient information content due to high rates of data removal. In this range, the classifier starts losing its discriminating ability. With regard to the increase in δ at 80%, the system starts behaving erratically, most likely as we are approaching the noise floor of the classification algorithm.

Furthermore, by examining the removal of common data with respect to performance of the classification algorithm, that is, the percentage of error. It can be observed that as the error rate increases the discrimination index also increases. The possible reason for this observation may be that the dataset used was small. Therefore, the overlap of common data points between the users is not notable as the data points are most likely to be far apart already thereby making the data removal to seem ineffective. The effects of the common data removal may be properly observed where a substantially large dataset is used as there would most likely be notable overlap of common data points from various users. In this case, the result would improve to show the effect of data removal on the error rate. Even though δ and δ_{min} are both maximised at 40% data removal (with somewhat an acceptable classification performance), a removal rate of 0% is optimal.

The top ten digraphs labelled as common by the OC-SVM at different data removal rates are presented in Table C1 of Appendix C. From observation, one can easily notice that type of digraphs that are typically flagged as common are those that include unprintable characters (characters that do not appear on the screen) such as `Backspace`, `Shift` and `Space`, to mention but a few. A possible explanation for these digraphs that include unprintable characters to be detected as common is that most users interact with the keys associated with these characters in very similar ways irrespective of their typing skill level. A similar observation was also made in [27].

Table 5.2: The effect of common data removal on the classifier's discrimination ability.

Data Removal Rate (%)	**Missed Classifications**	δ	δ_{min}	**% of Error**
0.00	0	1.502	0.746	0.00
20.00	0	1.698	0.829	0.00
30.00	1	1.752	0.794	2.00
40.00	1	1.912	1.125	2.00
50.00	5	1.588	0.859	10.00
60.00	7	1.421	0.710	14.00
70.00	11	1.319	0.656	22.00
80.00	19	1.357	0.635	38.00

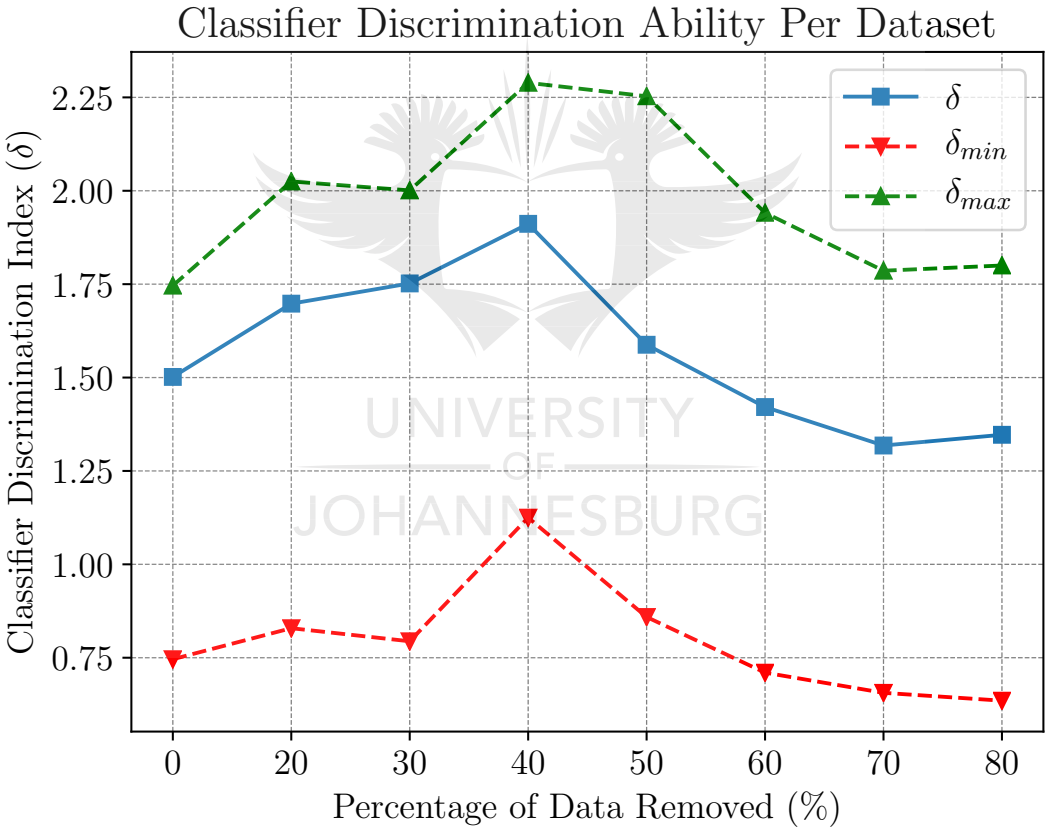


Figure 5.3: Classifier performance using the discrimination index to investigate the effects of removing common data.

In addition to digraphs that include unprintable characters there are digraphs that do not consist of unprintable characters that were identified as common. The possible justification for printable digraphs such as `in` and `nt` being classified as common is (application-

specific) or (domain-specific) as these digraphs appear in the keywords of the programming language used, those keywords include: `int`, `main` and `include`. Therefore due to repeatedly typing the keywords associated with these digraphs it is most likely that muscle memory comes to play thus resulting in almost everyone having similar timings for those digraphs. Moreover, these digraphs appear in a lot of words in the English language (which most participants use when typing), therefore, most users are accustomed to typing them which results in them being type at fast speeds which are comparatively similar for most users.

5.5 Remarks

While the proposed method is capable of detecting commonly occurring digraphs amongst users in a typist dataset it has some shortcomings which include.

1. Fitting the models is a memory intensive task (and is time consuming as well) and it gets worse as the value of ν increases. Scholkopfk *et al.* [50] report the correlation between the value of ν and the time taken to fit a model, it was found that ν is proportional to the memory used to store the model and the time taken to build it. The findings from our experiments corroborate the observations made by the authors of [50].

2. Applying the models at varying ν values to the dataset (in order to label the each data point as outlying or inlying) is a time consuming task.

3. The accuracy of the removed common digraphs relies heavily on the sample data used when creating the super user, therefore, information about other commonly occurring digraphs may be missed. To address this limitation, the number of samples used to create the super user may be increased (ideally, one would use all the keystroke data), however, it would be infeasible as it would require a lot of computing resources and time which were not at our disposal.

5.6 Conclusion

This chapter proposed the use of a novelty detection algorithm, the OC-SVM to eliminate commonly occurring typing traits amongst users in a typist dataset to improve classification accuracy. A brief theory of the components and concepts used to devise the proposed approach was presented– one-class classification and the OC-SVM. For the evaluation of the proposed algorithm, classification-based experiments were designed and

conducted and the obtained results discussed. Lastly, the chapter concluded with brief remarks– the strengths and shortcomings of the proposed approach.

Chapter 6

Application of Text Retrieval Methods in Keystroke Dynamics

This chapter[1] investigates whether text retrieval methods can be successfully employed for the task of person verification by keystroke dynamics. The remainder of this chapter is organised as follows. We start the chapter by exploring related works in both keystroke dynamics and text retrieval. We then present a brief review of text retrieval theory and its key concepts which include stop lists and term weighting, to mention but a few. Thereafter, we describe our authentication algorithm which applies text retrieval methods to achieve verification of online persons using a typing biometric.

6.1 Related Work

There exist two branches of KSD, namely, fixed-text and free-text keystroke analysis. In fixed-text KSD, keystroke analysis is performed on pre-defined fixed-length text [5]. In the case of free-text keystroke analysis, the user provides arbitrary text of variable length [54, 39]. When applying fixed-text keystroke analysis only static authentication can be achieved since analysis is limited to when a user enters their PIN/password which happens at the initiation of a computer session. On the contrary, using free-text keystroke analysis allows continuous authentication by analysing keystrokes provided beyond the login phase.

Most of the research done in KSD is in fixed-text analysis thus making the research in free-text analysis quite limited [8]. However, recent studies show that significant progress is being made in free-text analysis. The reason for this progress is that unlike fixed-text analysis which only limited to the initiation of a computer session, continuous authenti-

[1]Part of the work presented in this chapter was published in [53].

cation can be achieved beyond the login phase by employing free-text keystroke dynamics [55].

In the sections to follow, existing studies and commercial products that employ both fixed-text and free-text keystroke analysis will be reviewed. However, it should be noted that the discussions relating to fixed-text keystroke analysis will be brief and more emphasis will be placed on free-text keystroke analysis as it is the most relevant to this study.

6.1.1 Fixed-Text Keystroke Analysis

In KSD literature, fixed-text keystrokes analysis is usually employed to supplement knowledge-based authentication schemes such as PIN or Password in order to make them more robust; this case of fixed-text keystroke analysis is known as *password hardening*. The idea here (in password hardening) is that users type in their login credentials; viz. secret keys, usernames and names regularly thus making their typing signature in this case very unique compared to when they are typing anything else. This unique typing signature is typically accompanied by fast typing speeds. The reason for the fast typing and unique signature is mainly attributed to muscle memory– the user's muscles are used to the finger movements involved when typing in their password (learnt through repetitive typing of the password) therefore, resulting in very fast typing speeds even though the user may not be a fast/skilled typist.

With a password hardening system in place, when a user's password is compromised, the keystroke analyser acts as a second layer of authentication. Consequently, this implies that in addition to cracking the secret key, an impostor would also have to mimic how the legitimate user types their password which may not be an easy task. Some of the notable work done in password hardening are shown in Table 6.1. From the table, it can be observed that the researchers have suggested various classifiers which employ distance measures, classical statistical measures and machine learning algorithms.

Unlike password hardening, where verification is limited to the initiation of a computer session, there exist a branch of fixed-text analysis where identity verification can be achieved beyond the login phase. Here, the user is presented with a fixed-text prose to copy-type during both the enrolment and verification phases. The length of the typing samples are longer than those used in password hardening, usually an entire sentence or longer. At the authentication phase, the typing sample provided during verification is compared to the one collected at enrolment. Table 6.2 presents some of the notable

studies performed in this branch of static keystroke authentication.

Table 6.1: Password hardening algorithms and their performance.

Study	Sample Content	Classifier	IPR (%)	FAR (%)
Joyce & Gupta [56]	Username, password, names	Statistical	0.25	16.36
Bleha *et al.* [57]	Name and phrase	Bayes	2.80	8.10
Revett *et al.* [58]	Password	Distance	5.60	5.60
Yu & Cho [59]	Password	OC-SVM	0.00	0.30
Rodrigues *et al.* [60]	Numeric password	HMM	3.60	3.60
Brown & Rogers [61]	Name and surname	Neural Network	12.00	21.20
Ong & Lai [62]	Password	Clustering	15.00	15.00

According to our literature search, Gaines *et al.* [18] were perhaps the first to formally introduce the idea of user identification by means of keystroke dynamics. Gaines' experiment consisted of six professional secretaries who were seasoned typists. Each secretary was tasked to copy-type a prose of approximately 6,000 characters. This was done twice, with the second sample collected 4 months later. The digraph times of the recorded keystrokes were calculated and used as features. Through investigation, it was found that 87 digraphs appeared more than ten times in all the typing samples. After performing significance statistics, it was discovered that only 5 of the 87 digraphs were necessary to perfectly discriminate the typists (this implies an IPR and FAR of 0.00%). Those digraphs are: `in`, `io`, `no`, `on` and the fifth digraphs could either be `ul`, `il` or `ly`. Even though the keystrokes were logged on a computer, the classifications were performed by hand and no automated classification algorithm was employed.

Umphress and Williams [63] collected two typing samples from 17 participants who were tasked to copy-type prose. The first sample, was about 1,400 characters long and was used as a reference sample. The second sample, viz. the test sample, consisted of approximately 300 characters. The recorded keystrokes were then grouped into words and the first six digraph times for each word were computed and stored in a matrix. A statistical classifier was employed where a digraph time had to be within 0.5 standard deviations of its mean for it to be considered valid. For the decision rule, a threshold-based approach was followed. If the number of the valid digraph times a were greater or equal to the set threshold, the sample is classified as belonging to the particular user undergoing verification. The authors report an IPR of 6.00% and a FAR of 12.00%.

Bergadano *et al.* [64] propose a custom measure for keystroke dynamics that uses the tri-graph times as features. When comparing two samples, tri-graphs are extracted from the samples and common tri-graphs between the two samples are stored in separate lists ordered by their average durations (and alphabetical order is used as a tie-breaker). For a sample to be classified as belonging to a particular user, the verification sample must be closer to that of the user than that of any other user in the system. To evaluate their approach, 154 volunteers were asked to type a prose of about 683 characters. Of the 154 experiment participants, 44 acted as legal users providing five samples and the remaining 110 acted as impostors and only provided a single sample. The authors report that their algorithm achieved an IPR of 0.14% and a FAR of 0.00%.

Even though this method of authentication proved to be effective, it was not without limitations. The main limitation being that when the user is being verified their activities are abruptly stopped so that they copy-type the prose in order to be verified. This discontinuance negatively impacts the usability and user-friendliness of systems that employ such an authentication scheme as it disrupts the user's activities. To address the aforementioned limitation, researchers proposed a non-obtrusive identity verification approach which is presented in the immediate section to follow.

Table 6.2: Static authentication algorithms and their performance.

Study	**# Users**	**Sample Length**	**Classifier**	**IPR (%)**	**FAR (%)**
Gaines *et al.* [18]	6	6000	Manual	0.00	0.00
Umphress & Williams [63]	17	1400 + 300	Statistical	6.00	12.00
Bergadano *et al.* [64]	154	683	Degree of disorder	0.14	0.00

6.1.2 Free-Text Keystroke Analysis

This section presents some of the notable research done in continuous authentication research using free-text keystroke analysis. The studies which are discussed are summarised in Table 6.3.

Gunetti & Picardi (GP) [33] suggested a continuous verification approach based on the static typist verification work by Bergadano *et al.* [64] discussed in the previous section.

GP collected keystroke data from 205 participants over a period of 6 months. Of the 205 volunteers, 40 acted as genuine users and provided 15 typing samples whereas the 165 provided a single sample acted as impostors. The logging of the keystrokes was done via a web application that ran on a browser, so the data was typed into an HTML form and the keystroke timestamps were obtained using *JavaScript*. In this study n-graphs of latency times were used as features, specifically digraphs, tri-graphs and four-graphs which implies $n = 2, 3, 4$ respectively. The implemented authentication algorithm achieved IPR of 0.03% and a FAR of 3.17%. Even though the typing samples in this this study were written in Italian, GP's algorithm was also tested on English text samples and it was reported that there were insignificant changes in the algorithm's performance [54, 65].

Table 6.3: Free-text keystroke dynamics authentication algorithms and their performance.

Study	**# Users**	**Classifier**	**IPR (%)**	**FAR (%)**
Dowland & Furnell [66]	63	Statistical	4.90	0.00
Gunetti & Picardi [33]	205	Nearest Neighbour	0.03	3.17
Messerman *et al.* [67]	55	Nearest Neighbour	2.02	1.84
Stewart *et al.* [34]	30	k-Nearest Neighbour	-	-
Ahmed & Traore [35]	53	Neural Networks	0.015	4.82

Dowland and Furnell [66] gathered approximately 3.5 million keystrokes from 35 participants. Their data collection was carried out for a period of 3 months. They implemented an unrestricted data logger, this is, a data logger that records all keyboard activities across all applications on a computer. They implement a statistical classifier using digraph times as features; this classifier was taken from their previous study [68]. A digraph time is accepted if it is within a certain factor of its mean. When evaluated, their algorithm achieved an IPR of 4.90% and a FAR 0.00%. The authors reported that when the five worst users (viz. users with inconsistent typing patterns) were removed, the IPR reduced to 1.7%.

The work of Messerman *et al.* was based on the works by [64] and [33] with the aim of addressing some of the shortcomings of [33] such as the high FAR and the long decision-making time due to comparing a new sample with all user samples stored in the database. A web-mail application was implemented for the purpose of data collection and a dataset

of about 3 million keystrokes with 55 users was compiled over a period of 12 months. The results reported by the authors show that the decision-making time of the algorithm was reduced substantially[2] and the FAR was reduced, however, the IPR they achieved was more than the one achieved in [33].

Stewart *et al.* [34] investigated the use of keystrokes and stylometry traits for authenticating online test takers. The stylometry part of the system focused on the linguistic features in the student's text i.e. what the student typed. The dataset in this study was made using typing data from 30 university students taking an online exam. This system was built to improve the performance of an existing system. The verification system uses digraph times as a feature and a k-nearest neighbour based classifier. The system was tested in an open and closed-setting and achieved EERs of 1.4% and 0.55% respectively. This change in the EER shows how performance in a closed system is better compared to that of an open system. The authors reported that the KSD component performed better than the stylometry module. This study has the same aim as the study of this dissertation which is to employ KSD as method of verification in order to ensure academic integrity.

Ahmed & Traore [35] suggest an approach for continuous authentication using KSD that combines monograph and digraph analysis and employs a neural network to predict missing digraphs based on the relation between the observed keystrokes. In this study the keystroke logger was unrestricted. The hold-time and a digraph of latency times were used as features for building the typist profile. The authors claim that their approach achieves an accuracy level comparable to the state-of-the-art authentication algorithm at a very low processing time. This system was experimentally evaluated in both an open and closed environment using a dataset with 53 subjects. It was reported that the implemented algorithm achieved an IPR of 0.015% and a FAR of 4.82%.

The major challenges in free-text keystroke analysis include the lack of publicly available datasets and the lack of benchmarking studies to test the performance of authentication algorithms across all available datasets [55]. The aforementioned challenges can be attributed to the common practice in KSD research that each researcher or research group that publishes a study creates their own dataset and measures parameters that suit their application[3]. Thus, finding the state-of-the-art is a challenge due to the variations in data

[2]In GP's study, the reported decision-making time for their authentication algorithm is about 140 *s*. The authors of [33] report that they were able to reduce this time to 0.5 *s*.

[3]An example of this is GP's study were only `keydown` events and key codes were recorded. Not recording keyup events limits the usage of GP's dataset.

collection, and experimental and test procedure. However, most studies [69, 70, 71, 72] have replicated the study by Gunetti and Picardi [33] and used their dataset which makes their authentication algorithm one of the state-of-the-art in KSD authentication research.

The work presented in this chapter is inspired by some of the notable work done in computer vision as a result of the application of text retrieval methods to achieve object recognition in video content [73] and image searching [74]. Sivic and Zisserman [73] propose a text retrieval based approach to retrieve frames and snippets of a video containing a user specified object (user presents an image of the object). They obtain their training data by sampling several shots which represent about 10% of the frames in the movie. They extract local descriptors from the training frames and quantise the descriptor vectors using K-means clustering. They create the notion of a "visual" word which is defined as the cluster number, k, of the cluster nearest to the descriptor vector. In this view, an image can be seen as a text document made up of visual words. The visual words within the document are weighted according to the Term Frequency-Inverse Document Frequency (TF-IDF) score. Retrieval is then performed by calculating the cosine similarity between the query and source "documents".

Nister *et al.* [74] continue from the work of [73], however, unlike in [73] where a flat vocabulary is used they use a hierarchically defined vocabulary that forms a vocabulary tree. The vocabulary tree is constructed by hierarchically quantising descriptor vectors using hierarchical K-means clustering. In the case of the hierarchical version of the K-means, k defines the number of children of each node and not the final number of clusters. Instead of using the normal TF-IDF weighting scheme, they suggest hierarchical TF-IDF weighting for the hierarchically defined visual words in the vocabulary tree. The authors report that their approach allows for a more efficient lookup of visual words, the use of a larger vocabulary and improves quality of retrieval.

What is attractive with the employment of text retrieval methods in the aforementioned works is that it produced fast yet accurate results. This is because the matches are pre-computed. The authors in both [73] and [74] reported that they were able to query results on par as Google's text search in terms of the accuracy and speed. The authors in [74] state that the main drawback with the text retrieval approach is that it requires some pre-computations (building of the vocabulary of visual words) to be performed and the process is time consuming. This is why they ([74]) opted for the hierarchical version of the K-means instead of the flat version. Notwithstanding the aforementioned limitation, the fast querying and high accuracy makes it worthwhile to pursue this avenue.

6.1.3 Applications

Finding literature on real world applications that employ keystroke dynamics to enhance security was quite a challenge. However, there exist a few commercial products which are offered via the Internet as a software service. The challenge is that entities that own these applications do not release any information regarding implementation and performance of their systems in order to protect their trade secrets. *BioPassword Inc.* is a Software as a Service (SaaS) based company that owns the patent for the keystroke technology and provides software-based biometric security solutions [17]. Two other companies that provide keystroke dynamics based security enhancement solutions are *TypingDNA* [75] and *Deepnet Security* [76] with their solution named *TypeSense*.

6.2 Text Retrieval

Given a collection of documents, the task of text retrieval can be defined as the use of a user specified query to identify a subset of documents that satisfy the user's information need. In this section, we present the basic theory relating to text retrieval. We start this section by introducing the core terminology used in text retrieval literature which are paramount to our work [49].

- *Term*[4] - a sequence of characters that can be treated as a single logical entity. Terms are usually words, however, there exists terms that are not thought of as words, such as "MI6" or "K-9".
- *Bag-of-Words* - a model for extracting text features by ignoring the exact ordering of terms in a document but focusing on the number of occurrences of terms in the documents.
- *Document* - a unit of textual information.
- *Query* - a request for information.
- *Corpus* - a collection of all the documents on which text retrieval is performed.
- *Vocabulary* - a collection of all unique terms found in the corpus.

Text retrieval systems typically employ a standard procedure [49]. We briefly describe this process in the context of text retrieval using the English language. First, the text

[4] "Term" and "word" are used interchangeably by most authors, however, we shall use *term* for the sake of consistency, avoiding ambiguity and to minimize confusion.

in the documents is tokenised, converting the each document into a list of terms. Linguistic pre-processing[5] such as stemming is then applied to terms. Stemming represents each word with its stem, for example the words: "type", "typing" and "typed" would be represented by the stem "type". Thereafter, a stop list is then built and used to remove words that appear in most documents (these words are referred to as stop words) as such words are not discriminating for any particular document. Example stop words in English include "a", "is" and "the", to mention but a few.

The words that remain after the stop words removal are then assigned unique integer based identifiers and each document is represented by its vector equivalent known as a document vector. A document vector is a representation of a document as a vector with one component corresponding to each term in the vocabulary, together with a weight for each component [49]. There are various ways of weighting terms, we discuss them in Section 6.2.2. Finally, the set of document vectors representing all the documents in the corpus are organised as an inverted index/file. An inverted index is book-index like data structure with each term in the vocabulary together with a list of all the documents (and position in that document) in which the term appears [73]. The purpose of an inverted index is to allow fast full-text searches, at the cost of increased processing when a document is added to the database. What makes an inverted file the preferred approach over other possible data structures is its efficiency with respect to storage space and retrieval time. All the aforementioned processes are performed to construct the inverted index and they take place before the actual retrieval.

For the actual retrieval to take place, a user makes a query which details the information they wish to retrieve from the system. The user's query goes through the same pre-processing as described above in order to get its vector representation. Thereafter, the similarity (computed by measuring the angle between two vectors) between the query vector and each of the document vectors of the documents in the corpus is the computed, this is discussed in detail in Section 6.2.3. The search result is a list of documents ranked by similarity score, in descending order. A document is considered to be relevant if the user deems the information contained within it to be of value with respect to their information need.

[5]The linguistic pre-processing is not limited to stemming only, there are other pre-processing task that are applied.

6.2.1 Stop List

In Chapter 5, we suggested a method for detecting and removing common features that occur amongst most users in our typist dataset. This was done with the aim of improving the classifier's discrimination ability as these commonly occurring features provide little or no discrimination therefore making them useless. As mentioned before, in text retrieval there already exists a way of handling poor discriminators (the most and/or least occurring terms) in the corpus. This is achieved by the use of a stop list.

The typical way of determining a stop list is by calculating the total number of times each term, t, appears in the corpus. This is known as the *collection frequency* and is denoted by cf_t. After computing cf_t for each term, the terms are sorted based on their cf in descending order. The top u terms (most common terms) and/or the bottom m terms (least common terms) are then filtered out so that they are not included during indexing; $u, m \in \mathbb{N}$. The terms that exhibit reasonable discriminative power are those which are neither common nor rare. Later, in Section 6.2.2 we will see how term weighting results in the most/least frequent terms having little impact on document ranking.

Even though an ideal stop list is made of both the most and least occurring terms, most studies in literature make use of a stop list for dropping common terms which shows that generally emphasis is put on the removal of the commonly used terms. The use of a stop list results in smaller inverted lists as there are less terms to be indexed therefore reducing the amount of disk space needed to store the inverted file. Having a smaller inverted file allows for even faster retrieval times. Moreover, the removal of poor discriminators improves the retrieval accuracy of a text retrieval system.

6.2.2 Term Weighting

In the section immediately above, it was established that not all terms are equally important. In this section, we explore ways that can be employed to further assess the importance of each term in a document especially when considering relevance between a user query and a document during retrieval. This evaluation of term importance can be achieved through term weighting, which is the assignment of numerical values (weights) to terms in order to represent their importance in a document, based on the statistics of the occurrence of the term [49, 77]. By applying term weighting, the focus is put on the occurrences of a term in a document and the semantics of the language used in the document, that is, the exact ordering of terms is ignored completely. The approach of extracting textual features is known as the Bag-of-Words (BoW) model. When using

this model, documents "Bokang types faster than Dintle" and "Dintle types faster than Bokang" are identical as they would have the same BoW model.

The simplest way of assigning a weight to a term would be to let the weight to be equal to the number of occurrences of term, t, in document, d. This weighting scheme is known as *term frequency* and is denoted by $tf_{t,d}$. Usually $tf_{t,d}$ is normalised by dividing it by the total number of terms in d. The usage of tf as a weighting function presents one main limitation: all terms are considered equally significant when evaluating relevancy on a query. As a matter of fact certain terms have little or no discriminating power in determining relevance. For example, given a collection of documents on a subject such as medicine it is likely that the term "diagnosis" will appear in almost in every document thus making the term less discriminative. The tf weighting scheme does not take into account the discriminating power of the term.

To address the aforementioned limitation that the tf weighting scheme suffers from, the effect of terms that appear too often in a document such that they are insignificant when determining relevancy is suppressed by multiplying the term frequency with what is called *inverse document frequency* and is given by:

$$idf_t = \log \frac{N}{df_t}; \tag{6.1}$$

where N is the number of documents in the corpus and df_t is the document frequency of term t. The document frequency is defined as the number of documents in which term t appears. The idf is a weight indicating how widely a word is used across the corpus. Therefore, the idf of a rare term is high, whereas the idf of a frequent term is likely to be low. Combining the definitions of term frequency and inverse document frequency brings about the widely used term weighting scheme in text retrieval literature known as *term frequency-inverse document frequency*, denoted by $tf\text{-}idf_{t,d}$. Using this scheme, the weight assigned to term t in document d is given by:

$$tf\text{-}idf_{t,d} = tf_{t,d} \times idf_t \tag{6.2}$$

Put differently, $tf\text{-}idf_{t,d}$ assigns a weight to term t in document d that is [49]:

- highest when t occurs many times within a small number of documents, therefore, lending high discriminating power to those documents;
- lower when t occurs fewer times in a document, or occurs in many documents, therefore, offering a less pronounced relevance signal;
- lowest when t occurs in almost all documents.

6.2.3 Scoring and Ranking

In the final stage of the text retrieval process, documents that contain information that is similar to the query are identified. The similarity is determined by computing the distance between a (query, document) pair for each document in the corpus. The value returned by the distance function, known as the similarity score, is used to rank the search results (list of documents), with the most similar documents appearing at the top. In this section, we look at the mechanics involved when scoring and ranking documents to identify a group of documents that fulfill the user's information need.

In text retrieval, a document is represented by a vector with each component corresponding to each term t in the vocabulary, together with its weight w_{t_i} computed using some weighting function [77]. Assuming a vocabulary of L terms, each document d is represented by a document vector of length l given by

$$\vec{V}(d) = [t_0, w_{t_0}; t_1, w_{t_1}; t_2, w_{t_2}; ...; t_L, w_{t_L}]^\intercal \tag{6.3}$$

Vocabulary terms that do not appear in document d are assigned a term weight w_{t_i} of zero. The standard weighting function used in text retrieval is the TF-IDF, therefore, w_{t_i} is usually computed using (6.2). By converting each document in the corpus to its vector form, we end up with set of documents as vectors in a common vector space, in which each term has its own axis. This is known as the *vector space model*.

There a number of distance functions that can be employed for determining similarity measure, we only discuss the cosine similarity as it is considered the standard in literature. We denote the similarity between two documents, d_1 and d_2 as $sim(d_1, d_2)$. The cosine similarity between two documents is given by

$$sim(d_1, d_2) = \frac{\vec{V}(d_1) \cdot \vec{V}(d_2)}{|\vec{V}(d_1)||\vec{V}(d_2)|}; \tag{6.4}$$

where the numerator represents the dot product between the document vectors and the denominator is the product of their Euclidean lengths. The presence of the Euclidean lengths in the denominator length-normalises the document vectors $\vec{V}(d_1)$ and $\vec{V}(d_2)$ to unit vectors $\vec{v}(d_1) = \vec{V}(d_1)/|\vec{V}(d_1)|$ and $\vec{v}(d_2) = \vec{V}(d_2)/|\vec{V}(d_2)|$. Therefore, we can rewrite 6.4 as

$$sim(d_1, d_2) = \vec{v}(d_1) \cdot \vec{v}(d_2). \tag{6.5}$$

The similarity measure $sim(d_1, d_2)$ is the cosine of the angle θ between two document vectors. The value of θ is defined in the range $[0, \pi]$; therefore, $0 \leq sim(d_1, d_2) \leq 1$, with a score of 1 being the best and 0 is the worst. An illustration of the cosine similarity

measure is shown in Figure 6.1.

A user defined query q can be treated as a very short document. Therefore, a vector, $\vec{V}(q)$ can be derived from a user defined query, q; with $\vec{V}(q)$ being in the same vector space as the document collection. The similarity score between the query vector and each document vector in the corpus is then computed as described above and the results ranked by similarity score.

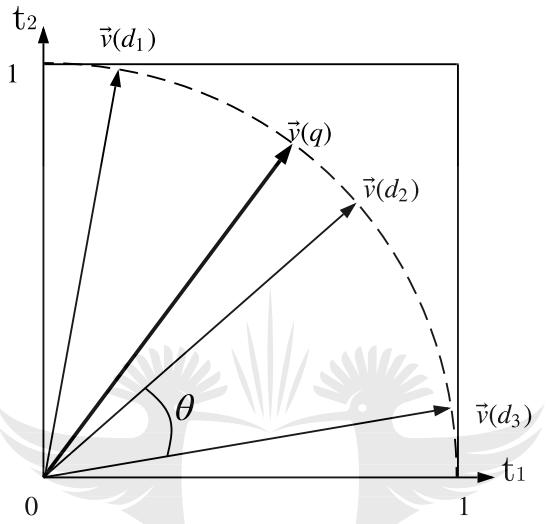


Figure 6.1: Illustration of the cosine similarity measure between two documents; $sim(d_2, d_3) = \cos\theta$.

6.3 Approach

In this section, we present our keystroke authentication algorithm that is based on text retrieval methods. We start by describing how the notion of a term is created in order to enable us to build word-like text features using keystroke features. Lastly, we describe the authentication algorithm used to verify users.

6.3.1 Building the Vocabulary

In order to apply text retrieval methods, we build a vocabulary of terms using keystroke features similar to how [73] and [74] create the notion of visual words. The aim here is to quantise the timing information associated with a digraph sequence (fk^i,tk^j) into several clusters. We then use the (fk^i,tk^j) pair and the cluster number to create a term in the form "$fk^i\_tk^j\_k$"; where k is the cluster nearest to the timing information of digraph

(fk^i, tk^j). The quantising of the timing information is achieved by employing K-means clustering. The process of building a vocabulary of terms is as follows.

1. Select all keystroke data associated with digraph sequences that have at least hundred occurrences.

2. Perform feature extraction and pre-processing on the sampled keystrokes as described in Chapter 4.

3. For each unique digraph sequence, apply K-means[6] clustering on the PCA generated timing features associated with the sequence to obtain cluster centroids associated with timings for each digraph sequence. The K-means clustering algorithm was initialised as follows:

   - Initialisation method: k-means++. Selects initial cluster centers in a way that ensures fast convergence.
   - Number of clusters (k): 8. The number of clusters to form as well as the number of centroids to generate. Therefore, $k \in [1, 8]$.
   - Number of jobs: -1. Used to specify how many concurrent processes should be used for parallelised routines. If set to -1, all CPUs are used to concurrently run computations.

4. Create a lookup table of all the digraph sequences and their K-means cluster centroids.

5. To create terms, iterate through each user's keystroke data and for each keystroke extract the digraph sequence and look it up in the lookup table.

   - If the digraph sequence is present in the lookup table, compute the Euclidean distance between the user's times for the digraph and the sequence's centroids. Return the cluster number, k, of the closest centroid.
   - If digraph sequence is not present in the lookup table, it is ignored.

By building a vocabulary of terms, we have converted keystroke dynamics features into "textual" features or words, therefore redefining authentication by keystrokes dynamics as a text retrieval problem. We can now create a clear analogy between keystrokes dynamics and text retrieval as follows:

- keystroke features are terms,

[6] A public implementation of this algorithm by Sci-kit learn was used, see [52] and https://scikit-learn.org.

- a user is equivalent to a document,
- a typing sample provided during authentication is a query.

6.3.2 Authentication

The final and maybe the most important stage in the authentication process is where the system makes a decision to accept or reject the identity claim made by the user by means of some predefined decision rule. This rule plays a vital role for determining the overall robustness of the system. If the rule is too lenient, easy access is given to everyone including impostors (resulting in a high IPR) and if the rule is too strict, it may be very robust against impostors but wrongfully reject legitimate users too often (resulting in a high FAR). Therefore, this rule has to be devised to ensure an acceptable trade-off between the IPR and FAR. In this section, the process of authenticating a subject using the suggested authentication algorithm is described.

Assume a new typing sample q_X (that is claimed to belong to user U– a known user) is presented to an authentication system. The system must determine the validity of the claim. It is possible that q_X may belong to U or another known user or someone who is unknown to the system. To determine the validity of the claim, the following process is followed:

1. Compute the cosine similarity between query q_X and the reference document of U; viz. $s_U = sim(q_X, d_U)$.
2. Compute the cosine similarity $sim(q_X, d_n)$ between q_X and all the documents in the collection, i.e. d_1 through d_n; where n is the number of users in the system.
3. Compute the average of the similarity scores (s_U is excluded when computing the average) of all the (q_X, d_n) pairs which is given by:

$$s_{avg} = \frac{1}{n-1} \sum_{i=1}^{n} sim(q_X, d_i) \quad i \neq U \tag{6.6}$$

4. The system then accepts or rejects the identity claim using the following rule:

$$h(s_U) = \begin{cases} accept, & \text{if } s_U/s_{avg} > T \\ reject, & \text{if } s_U/s_{avg} < T \end{cases}; \tag{6.7}$$

where $h(s_U)$ is the decision function and T is a fixed preset threshold. The threshold T is determined experimentally and may vary from one dataset to another. Its formulation is discussed in Section 7.2.1.

6.4 Conclusion

This chapter investigated the application of text retrieval methods and concepts for user verification by keystroke dynamics. The related works in both text retrieval and keystroke dynamics were explored. A brief discussion of text retrieval concepts that are important to the proposed algorithm was also given. To end the chapter, the proposed keystroke authentication algorithm based on text retrieval methods was presented.

Chapter 7

Experiments

A system that performs keystroke analysis can be evaluated on three tasks when a new sample X is presented to the system:

- **Classification**: X is from a one of the known users. The system must determine who actually provided the sample.
- **Authentication**: X is claimed to belong to user U (a known user). The system must determine the validity of the claim. It is possible that X may belong to U or another known user or someone who is unknown to the system.
- **Identification**: X is presented to the system. The system has two possible responses: (a) X belongs to user U; (b) X belongs to someone unknown.

In this chapter, we conduct experiments to evaluate the performance of the suggested text retrieval based keystroke analysis algorithm on two of the aforementioned tasks, namely, classification and authentication. Furthermore, with the aid of experiments, the behaviour of the suggested algorithm will be studied with respect to the removal of common data and the query length. This done for both classification and authentication tasks. This will help determine the optimal parameter settings that can be used to achieve accurate results. Lastly, the approach suggested in this study is compared with one of the state-of-the-art KSD authentication algorithms.

7.1 User Classification

To conduct the experiments detailed in this section, documents with at least 2000 terms are selected. Thereafter, each document is divided into two sets, the first set is used as the reference document and it remains in the corpus. The second document which consists of 1000 terms is used a query. For each query drawn from each of the n users, the

cosine similarity $sim(q, d_n)$ is computed between q and all the documents in the corpus, viz. d_1 through d_n; where n is the number of users selected for the experiment. Ideally, the classification algorithm should assign the highest similarity score to the (q, d_n) pair that were drawn from the same user and lower scores for the rest of the pairs. If this is the case, it is recorded as correct classification and if the opposite is true then a missed classification is recorded.

In addition to using our datasets, the algorithm is evaluated on two more datasets, namely, Villani's [78, 79] and Stewart's [34, 80]. Villani's dataset was collected from a diverse pool of subjects which include university, students, faculty staff, friends and family resulting in 144 participants in total. The participants were given typing tasks for both free-text and fixed-text input. For the free-text collection, participants were instructed to respond to open-ended essay questions and for fixed-text collection they were required to copy type fables. Furthermore, the users were required to perform the same typing tasks on a desktop keyboard and on a laptop PC keyboard. We treat the data as separate datasets as it was collected under different environmental conditions. Stewart's dataset was gathered from 43 students taking a spreadsheet modelling course. The keystroke data was logged on free-text input in an examination room (a computer laboratory equipped with the same desktop computers) while the user was attempting assessment questions. A numerical summary of the data collected by Villani[1] and Stewart is given in Table 7.1.

To quantify the performance of the system, we report the number of missed classifications and the system's percentage of error.

Table 7.1: Data collection summary of third party datasets.

Dataset	Users	No. Keystrokes	Avg. Keystrokes/User
Villani (Desktop)	105	360,000	3,400
Villani (Laptop)	87	300,000	3,400
Stewart	43	780,000	18,000

7.1.1 Results

The results of the user classification experiment are reported in Table 7.2. Looking at the table, it can be observed that the algorithm performed well on most datasets, how-

[1]Villani's dataset consists of data collected on both fixed and free-text input, we only report numbers related to the collection of keystrokes on free-text input.

ever, the worst performance overall was obtained on Stewart's dataset where six missed classifications were recorded.

Table 7.2: Classification performance of our algorithm on multiple datasets. The number of classifications made is 43, which is the number of users selected in each of the datasets.

Dataset	**Missed Classifications**	**% of Error**
Stewart	6	13.95
Villani (Laptop)	2	4.47
Ours (Assessment)	1	2.33
Villani (Desktop)	0	0.00
Ours (Practice)	0	0.00

In order to determine the possible reasons that may have resulted in the poor performance of the algorithm on Stewart's dataset, a further analysis was performed. This was done by plotting the colourmap shown in Figure 7.1 which shows the cosine similarity score between every (q, d_n) pair by colour intensity (dark shade of gray for high sim scores and lighter shades for low sim scores) and the missed classifications which are marked by a red circle around the cell. The cells that lie on the main diagonal represent the sim scores obtained on (q, d_n) pairs that belong to the same user. Ideally, the similarity scores on the main diagonal should be the highest in their corresponding columns thereby resulting in the cells on the main having a deep shade of gray and all the other surrounding cells having lighter shades of gray.

By paying attention to the shades of cells that were incorrectly classified and comparing them to the shade of the cell on the main diagonal in the respective column, it can be observed that the shading of the cells is very similar. This means that the similarity scores in this case are quite close which implies that the two users (the actual users and the incorrectly classified user) may exhibit similar typing behaviour. Based on the observations made, we anticipate that the missed classifications that were observed when testing on Stewart's dataset are most likely to have occurred on users that have indiscriminate typing traits (as shown by the very light shading of the cells on the main diagonal) which leads to poor self-matching when classifying samples.

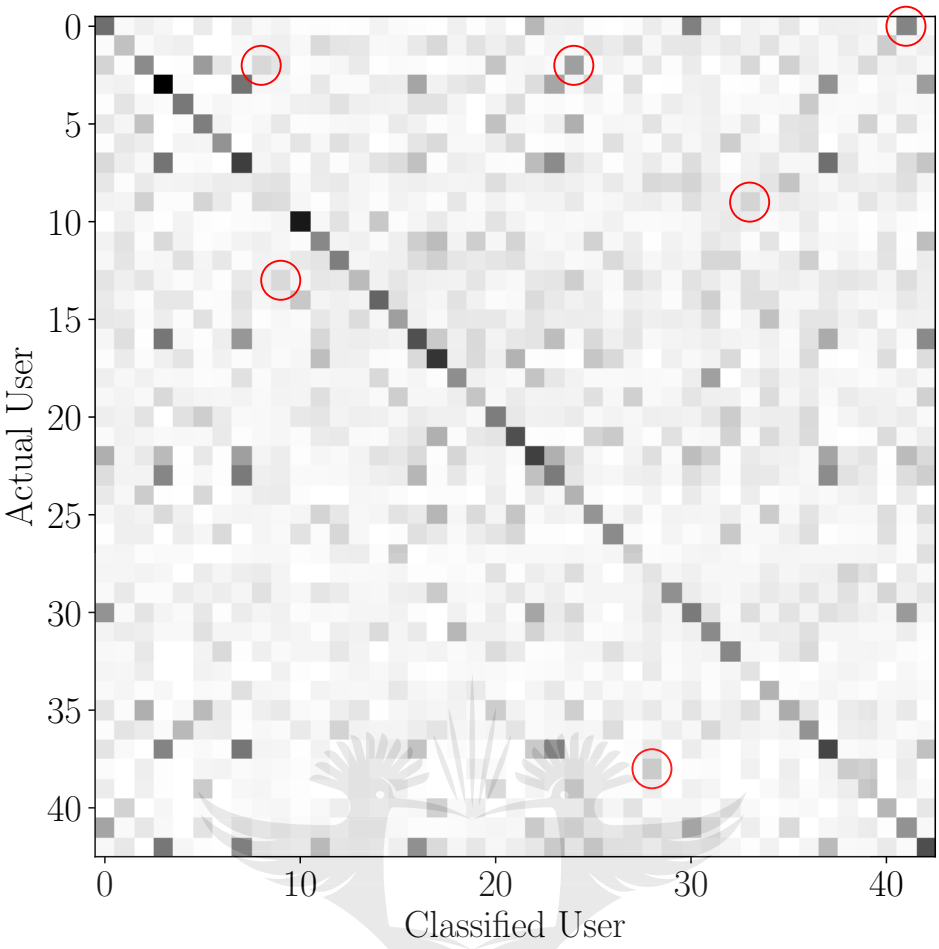


Figure 7.1: Colourmap of cosine similarities scores between the actual users and the classified users for Stewart's dataset. Each cell represents a similarity score between a (q, d_n) pair. Cells that lie on the main diagonal represent the scores obtained on (q, d_n) pairs that belong to the same user and the red circles around the cells represent missed classifications, i.e. a (q, d_n) pair that scored more than that of the actual user.

7.1.2 Environmental Effects on Typing Behaviour

Considering that in this study the users' typing behaviour was monitored in two different environments, i.e. the assessment site and the practice site, we investigate whether a user's typing behaviour is consistent regardless of the change in environments. To achieve this, typing samples collected on the same site are compared against each other (the document collection/corpus and the query are from the same site). We also compare samples collected on the different sites against each other by using a corpus from one site and draw the query from a different site (e.g. corpus from the practice site and a query from the assessment site).

The authors of [78] conduct a similar investigation where their algorithm was trained and tested on data collected using the same keyboard (e.g. train and test on data collected using a laptop keyboard). The algorithm was also trained using data gathered using desktop keyboards and tested with data collected on laptop keyboards and vice versa. Therefore, we run this experiment on their datasets using our algorithm. In this experiment we only select users that have provided samples in both environments.

7.1.2.1 Results

The results obtained in this experiment where user typing samples collected in two different environments were compared to determine whether typing behaviour varies with a change in environmental conditions are shown in Table 7.3. It can be observed that when comparing samples obtained in the same environment the algorithm performs well in classifying users. This is the case for our datasets and those of Villani. When comparing samples from the same environments, the worst performance resulted in an error of 4.65% due to a misclassification of two users. When comparing samples from different environments, the performance dropped drastically especially on the Villani datasets. In this case, the classifier failed dismally at classifying users which is most likely due to the susceptibility of the users typing signature to the environment. This susceptibility in the case of our datasets may be due to some of the following factors:

- The change in hardware, the hardware used in assessments is the same for everyone. However, in the practice site a student could use multiple devices which include desktops, laptops, tablet PCs and mobile phones.
- On the practice site students can help each other which introduces typing behaviour of other individuals (known or unknown) thereby contaminating the typing profile.
- Students are usually relaxed when doing tasks on the practice site, however, during assessments they are likely to stress due to pressure which may cause their typing signature to change.

A similar trend can be observed when looking at Table 7.4, where the authors of [78] performed a similar experiment using their algorithm on Villani's datasets. From their results, it can be easily noticed that their algorithm performed better than our algorithm on both Villani's datasets. Though the differences are not significant for the same environment tests they are quite significant when samples from different environments are compared.

Table 7.3: Classification performance of our algorithm in varying environmental conditions. We test our algorithm on Villani's Desktop and Laptop datasets. The number of classifications made is 43, which is the number of users selected in each of the datasets.

A. OUR DATASETS			
Corpus	**Query**	**Missed Classifications**	**% of Error**
Practice	Practice	0	0.00
Assessment	Assessment	0	0.00
Assessment	Practice	18	41.86
Practice	Assessment	23	53.49
B. VILLANI'S DATASETS			
Corpus	**Query**	**Missed Classifications**	**% of Error**
Desktop	Desktop	1	2.33
Laptop	Laptop	2	4.65
Desktop	Laptop	28	65.12
Laptop	Desktop	34	79.07

In [78], the authors also investigate the consistency of a users typing behaviour by varying two independent variables- keyboard type (desktop keyboard and laptop keyboard) and input type (free-text and fixed-text). The users typing samples collected in different environments are then compared to see to test for consistency. The authors report high accuracies in cases where the training and test data we gathered under the same environmental conditions. On the contrary, when comparing samples collected in different environments the performance drop drastically; part of the results they reported are shown in Table 7.4. We only show the results related to experiments performed on free-text input as we also focus on free-text in our study.

The findings of [78] corroborate our findings regarding a users typing behaviour under varying environmental conditions. Based on the results, we are of the view that keystroke data used to train and test should be collected in a setting where there are minimal changes especially in terms of the hardware used, type of input and environmental conditions. Furthermore, we reckon that these inconsistencies in typing behaviour are most likely to worsen if the user is not a skilled typist, as in the case of our subjects.

Table 7.4: Classification performance of Tappert's algorithm in varying environmental conditions. The results are taken from [78].

A. SAME ENVIRONMENT				
Train	**Test**	**36-Subject**	**Full-Subject**	
		% of Error	**Subjects**	**% of Error**
Desktop	Desktop	1.70	93	6.70
Laptop	Laptop	0.50	47	2.10
B. DIFFERENT ENVIRONMENTS				
Train	**Test**	**36-Subject**	**Full-Subject**	
		% of Error	**Subjects**	**% of Error**
Desktop	Laptop	41	40	40.90
Laptop	Desktop	39	40	37.60

7.1.3 Removal of Common Data

In Chapter 5, we presented a OC-SVM based method for detecting and removing common typing patterns which can be applied at the data pre-processing step. It was shown experimentally that this pre-processing step improved the discrimination ability of a classifier thus making it a necessity. We perform a similar experiment here by exploiting the concept of a stop list which is found text retrieval literature. In this experiment, we investigate the effect of the most common/rare terms (stop words) on the classifiers discrimination ability. To achieve this, the size of the stop list is varied by increasing or decreasing parameters df_{min} (for rare terms) and df_{max} (for common terms) to reject terms that appear and/or do not appear in a certain percentage of the documents in the corpus.

Figure 7.2 shows the relationship between a term's usage in the corpus and its importance to the document in which it appears. In the figure, it can be seen that terms that occur very frequently (with a $df_t > df_{max}$) or very rarely (with a $df_t < df_{min}$) in the corpus have relatively low importance compared to terms that with a df that lies between df_{max} and df_{min} (these are the terms we would like to keep). Therefore, when indexing vocabulary terms, terms that have a document frequency df strictly higher than the given threshold which we denote as df_{max} are ignored. For example, if $df_{max} = 0.1$ all terms that appear in more than 10% of the documents are ignored meaning that only terms that appear in 10% of the documents or less remain. In this experiment, we only focus on the removal of common terms and not that of rare terms, therefore, we only vary df_{max} while df_{min} is

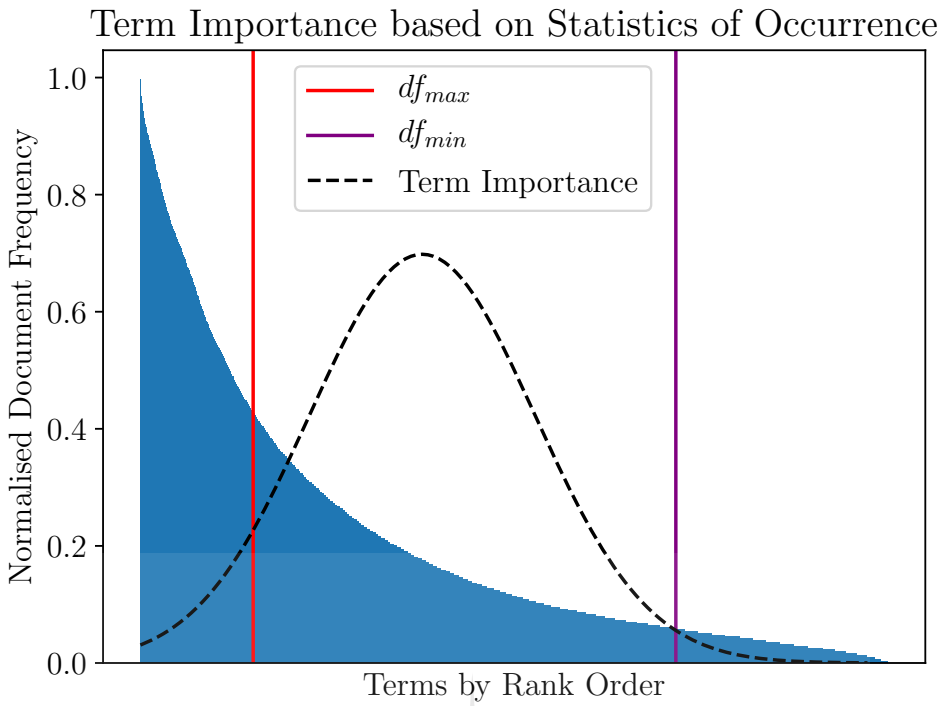


Figure 7.2: Plot shows the relationship between a term's importance (to a document it occurs in) and its statistics of occurrence in the corpus. A rare term has a $df_t < df_{min}$, a frequently used term has a $df_t > df_{max}$ and a term that is regarded as a good discriminator has a $df_{min} < df_t < df_{max}$.

always kept constant. The df_{max} was varied in the range [0.1, 1] with increments of 0.1 and a constant query length of 1000 terms was used.

In addition to missed classifications and percentage of error, we report the mean discrimination index (δ), minimum discrimination index (δ_{min}) and the percentage of the actual data removed due to the elimination of common terms. The discrimination index is computed as described in Section 5.4.

7.1.3.1 Results

The results obtained by performing the common data removal by means of a stop list are shown in Table 7.3. To improve the quality of results, df_{max} was incremented in steps of 0.025 in the range $[0.1, 0.2]$. This was done to improve the resolution in this range as the percentage of error associated with the classifier changes substantially between 0.1 and 0.2.

Observing the results presented in Table 7.5 and Figure 7.3, it can be seen that the effect of the removal of common terms on the classifiers performance is most visible when

Table 7.5: The effect of common data removal on the classifier's discrimination ability and overall performance.

df_{max}	**Missed Classifications**	δ	δ_{min}	**% of Data Removed**	**% of Error**
0.1	34	1.756	0.000	56.81	12.01
0.125	19	1.851	0.246	49.95	6.71
0.15	9	1.892	0.480	44.55	3.18
0.175	4	1.913715	0.753	40.38	1.41
0.2	2	1.913849	0.854	36.68	0.71
0.3	1	1.836	0.934	25.85	0.35
0.4	0	1.775	1.041	18.40	0.00
0.5	0	1.699	1.076	13.53	0.00
0.6	0	1.634	1.098	9.95	0.00
0.7	0	1.573	1.160	7.16	0.00
0.8	0	1.511	1.181	4.25	0.00
0.9	0	1.449	1.179	2.01	0.00
1.0	0	1.373	1.115	0.00	0.00

df_{max} is between 0.1 and 0.2, reason being that within this interval a substantial number of terms are removed. The terms that remain in this case are most likely rare terms that are unique to very few documents thus resulting in most documents being classified incorrectly. The overall trend between the size of the stop list and the classifier's performance suggests that the percentage of error associated with the classifier improves with a decrease in the size of the stop list.

The most important relationship in this experiment is that between df_{max} and the discrimination indices, i.e. δ and δ_{min}. These indices provide insight on how the classifier's discrimination ability changes with the removal of terms. By observing the δ and δ_{min} with relation to df_{max}, one can easily see that the discrimination indices increase until they peak at particular df_{max} values and then decrease thereafter. The δ is maximised at approximately $df_{max} = 0.2$ where terms that appear in more than 20% of the documents are removed thus resulting in an overall data removal of approximately 37%. This means that at $df_{max} = 0.2$ the classifier discriminates majority of the users well. This good discrimination ability may be attributed to the presence of mostly unique data in the user documents as most common data is removed at this df_{max} value. For any $df_{max} > 0.2$, the δ decreases. The possible reason for this observation might be that as df_{max} increases,

the amount of common data removed reduces thereby leading to documents consisting with substantially common terms which then decreases δ.

Even though δ gives us an indication of how well all the users are being discriminated, the δ_{min} give us insight on how well the worst user is performing (in terms of the discrimination index). Therefore, the optimal df_{max} is found where the δ_{min} is maximised. The maximisation of δ_{min} happens at $df_{max} = 0.8$. Another important observation to make is that at $max(\delta)$ two missed classifications are recorded and at $max(\delta_{min})$ however, the system achieves perfect classifications. This observation follows the saying: "*the chain is only as strong as its weakest link*", therefore, the optimal df_{max} is where δ_{min} is maximised. This maximisation of δ_{min} happens at $df_{max} = 0.8$.

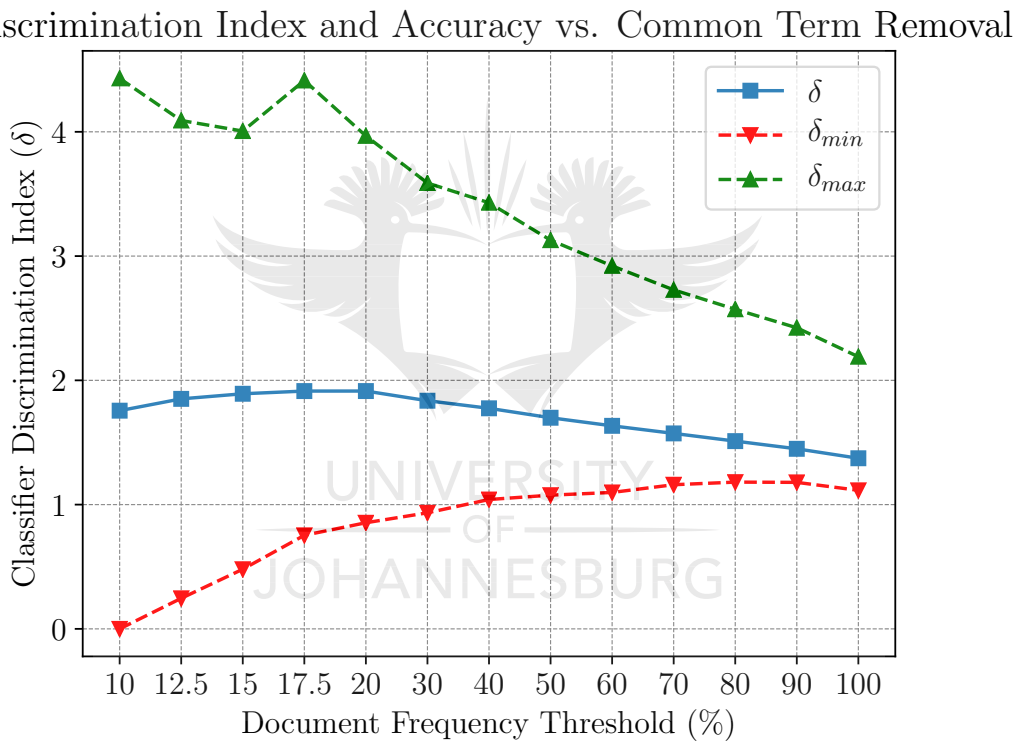


Figure 7.3: Plot shows the effect of common data removal on the classifier's discrimination ability and the overall accuracy of the classifier.

The following digraphs are the top ten digraphs flagged as common using the stop list (in descending order): `Backspace Backspace`, `in`, `Backspace Space`, `Backspace t`, `t Space`, `Tab Backspace`, `nt`, `Shift 9` (to produce '('), `Shift [` (to produce '{') and `Shift 5` (to produce '%') for $df_{max} \in [0.1, 0.9]$. At $df_{max} = 1.0$, no common data is removed as there no terms that appear in all the documents. Furthermore, it is worth noting that due to how the stop list works, the removed common terms are consistent for

all $df_{max} \in [0.1, 0.9]$.

When observing the experimental results obtained using the OC-SVM based method (in Chapter 5) and those presented here, it is clear that both sets of results are consistent. Of importance is that using both methods the mean discrimination index peaked at similar/close data removal rates, i.e. approximately 40% for the OC-SVM method and 37% for the stop list. These data removal rates are on par with each other as the difference between them is not substantial. Furthermore, both common data removal methods removed data associated with more or less the same common digraphs. The digraphs that appear in the top 10 commonly used digraphs for both algorithms are: `Backspace Backspace`, `in`, `nt` and `t Space`.

Based on the discussions, it is clear that both data removal methods produced consistent results (to a certain degree). However, it can be deduced that removal of common data by means of a stop list is the better option compared to the OC-SVM based approach, because:

- even though building the vocabulary is a time consuming task (due to the K-means clustering), it only happens once. Therefore creating a stop list is less computationally expensive and less time consuming compared to the OC-SVM based method where multiple OC-SVM models that correspond with the various data removal rates have to be created;
- the stop list method removes data with high accuracy and precision. The reason for the good performance by the stop list method is that unlike the OC-SVM method it does not used sampled data, rather, it computes the statistics of occurrence of terms using all the available data.

The findings obtained from performing this experiment further display the benefit of using text retrieval methods. The observations made here corroborate the findings made in Chapter 5 and further prove that the removal of commonly occurring terms does indeed improve the classifiers discrimination ability.

7.1.4 Query Length

The accuracy of a text retrieval system is, of course, related to the length of the query which we denote by $|q|$, i.e. the number of terms in the query. In general, the longer the query the better the chances of identifying documents that best match the user's information need, however, a text retrieval system should provide accurate results even

if the user query is not very long. Therefore, in this experiment, we investigate the effect of query length on the system's performance. Moreover, we aim to determine the least number of terms (and subsequently the least number of keystrokes) required to accurately authenticate a user. To achieve this, the length of a query is varied while df_{max} is kept constant at 0.8. The length was varied using a step size of 20 in the range $[20, 100]$ and a step size of 100 in the range $(100, 1000]$.

7.1.4.1 Results

Looking at the results presented in Table 7.6, it can be deduced that the classifier's performance improves with an increase in the query length. When using short queries, i.e. less than 100 terms, the classifier performs poorly. The reason for this is that when a query has too few terms its vector representation it will contain a lot of zero terms thus making it very sparse.

Table 7.6: The effect of query length on the classifier's performance.

$\|q\|$	**Missed Classification**	**% of Error**
20	167	59.01
40	86	30.39
60	48	16.96
80	28	9.89
100	8	2.93
200	1	0.35
300	1	0.35
400	0	0.00
500	0	0.00
600	0	0.00
700	0	0.00
800	0	0.00
900	0	0.00
1000	0	0.00

Acceptable classification accuracy is achieved using a query with 200/300 terms as only one missed classification is recorded for both query lengths. This means that with as little as 200 terms a user can be correctly classified 99% of the time. The actual number of keystrokes required to create J terms is given by $J + 1$, therefore, when using query with 200 terms the user needs to provide 201 keystrokes. If one desires even better accuracy,

then picking query lengths greater than 300 would be advisable. The effect of query length on the classifier's performance becomes invisible as the number of terms in the query increases (query length greater than 300), this is observed by the saturation of the classifier for query lengths greater than 300.

7.2 User Authentication

In this section, we perform experiments to understand the behaviour of the our approach when given the task of authentication which is not as easy as the classification task. Here, we perform experiments that test how well the system gives access to legitimate/legal users and how robust it is against impostors. Therefore, we perform two tests, namely, the legal connections and impostor attacks. For the legal connections test, a query known to belong to user U, q_U, is presented for authentication and the claimed identity is that of U. Ideally, the system should grant access since it is a fact that q_U belongs to U, however, if this is not the case a false alarm is recorded. The number of legal connections that can be attempted is equal to n, the number of users in the system. Therefore the number of legal connection we can attempt is 283 because as our system consists of 283 users. To quantify the performance of the system for the legal connections test we report the FAR which is given by #false rejections/#legal connections.

To test the robustness of the system against impostors, a legal user U is attacked using typing samples from all other $n-1$ users in the system. When a typing sample q_X (belonging to a known user U_X) is used to attack U, U_X's typing profile is temporarily removed from the system so that U_X is completely unknown to the system. This done to so to mimic a real world impostor attack. Ideally, when U is being attacked by all the $n-1$ users the system should reject all the identity claims. In the case where the system fails to reject an impostor, an impostor pass is recorded. This experiment is performed for every legal user in the system, therefore, the total number of attempted impostor attacks is $283 \times 282 = 79,806$. To quantify the algorithm's robustness against impostors we report the IPR which is given by #impostor passes/#attempted attacks.

For some of the experiments in this section, we use the knowledge learnt about the algorithm in the classification-based experiments such as the optimal stop list size and query length as (starting point) or (initial) values in the quest of determining the optimal parameter settings of the algorithm for performing the task of user authentication.

7.2.1 Decision Threshold

In Section 7.2, we presented a test that would be used to evaluate a user's identity claim during authentication. The test uses a threshold-based decision function to accept/reject users. In this section, we perform an experiment to determine the value of the decision threshold T by conducting the following experiment. A classification matrix similar to Figure 7.1 is computed using $df_{max} = 0.8$ and a query size of 1000 terms. Each similarity score that lies on the main diagonal is then divided by the average of the similarity scores in the respective column (the score on the main diagonal is excluded when computing the average); we term this quotient the *legal user quotient*, LUQ for short. Thereafter, the second best similarity score in the column is divided by the average of the similarity scores (excluding the similarity score on the main diagonal and the second highest sim score in the column) that remain in the column; we term this quotient *potential impostor quotient*, PIQ for short.

The LUQ gives insight on how a user's typing behaviour is distinctive compared to other user's in the system. If it is high, it means that the user has distinctive typing traits and if it is low, it means that the user possess indistinct typing traits. Therefore, a user with a low LUQ is prone to impostor attacks (due to them exhibiting indistinct typing behaviour) compared to a user with a high LUQ. On the other hand, the PIQ tell us the likeliness of a user being able to attack another user in the system due to the similar typing traits exhibited by both users (the legal user and the potential impostor). Therefore, if a user has distinctive typing behaviour (which results in a high similarity score compared to all other (q, d) pairs), the LUQ will be higher than the PIQ of the other user that is most similar.

The LUQ and PIQ values for all users are then presented using histograms as shown in Figure 7.4. Ideally, the two histograms should not overlap, in this case the FAR and IPR would both be zero. In reality, these two histograms are most likely to overlap as it has already been established that some users exhibit similar typing behaviour. Referring to Figure 7.4, it can be observed that the LUQ scores associated with users who have distinctive typing features occur high up in the blue histogram where there is little or not overlap with the possible impostors histogram. The opposite is true for LUQ scores associated with users who do not possess distinctive typing traits as it can be seen that there is overlap between the LUQ and PIQ histograms. This means that users within this region of overlap are more prone to intrusion attacks. Therefore, in terms of the overall performance of the system the saying– "*the chain is only as strong as its weakest link*" applies.

Moving the T to the right increases the FAR and decreases the IPR and moving it to the left does the opposite. Therefore, a T value that provides an acceptable trade-off between the FAR and IPR should be established. To determine the optimal value of T, we vary the value of T in the range $[1, 3]$ in increments of 0.2 and the T value that results in the least FAR and IPR is chosen.

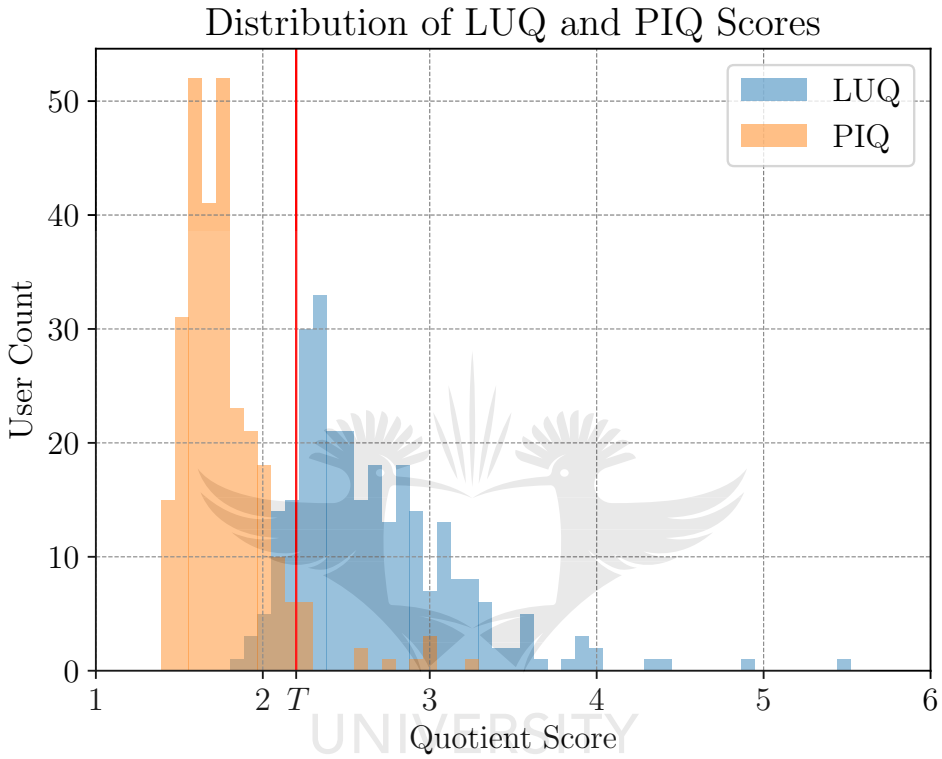


Figure 7.4: Histogram plot showing the distribution of LUQ and PIQ scores for all users in the dataset.

7.2.1.1 Results

The experimental results are presented in Table 7.7 and illustrated in Figure 7.5. It can be easily observed that for any $T < 1.8$ the best possible FAR is obtained as there no legal users who are falsely rejected, however, for the same T values the IPRs are at their worst. This observation shows that the T values in this range ($T < 1.8$), result in very lenient decision functions which give easy access to both legal users and impostors. As T increases more promising results are obtained and the optimal T value (where the trade-off between variables FAR and IPR is reached) is obtained at the intersection of the two plots which is approximately at $T = 1.9$ (the midpoint of values 1.8 and 2.0).

Table 7.7: Experimental results for authentication accuracy for different thresholds.

T	**Legal Connections**		**Impostor Attacks**	
	False Alarms	**FAR (%)**	**Impostor Passes**	**IPR (%)**
1.0	0	0.00	39811	49.88
1.2	0	0.00	19292	24.17
1.4	0	0.00	6813	8.54
1.6	0	0.00	2020	2.53
1.8	1	0.35	651	0.82
2.0	6	2.12	263	0.33
2.2	34	12.01	122	0.15
2.4	112	39.58	57	0.07
2.6	156	55.12	27	0.03
2.8	198	69.96	15	0.02
3.0	226	78.95	8	0.01

In order to obtain the actual FAR and IPR that can be achieved at the optimal T value, the experiment was performed for $T = 1.9$. From the experiment, it was discorvered that at $T = 1.9$ a FAR of 1.41% was recored as a result of four users being falsely rejected and the IPR was 0.57% due to 456 impostor passes. From the plot in Figure 7.5 it is quite obvious that at the point of intersection the FAR increases from what it was at $T = 1.8$ and on the other hand the IPR decreases even further. Even though $T = 1.9$ is the theoretical optimal point, we choose the optimal T value to be at $T = 1.8$ reason being that the FAR achieved at $T = 1.9$ is greater than 1% and at $T = 1.8$ both the FAR and IPR are less than 1%. When using the decision function associated with $T = 1.8$, a legal user will be falsely rejected 7 times out of every 2,000 authentication attempts and an impostor will be given access 41 times for every 4,500 attempted attacks. As T moves more to the right ($T > 1.8$), the IPR improves while the FAR worsens.

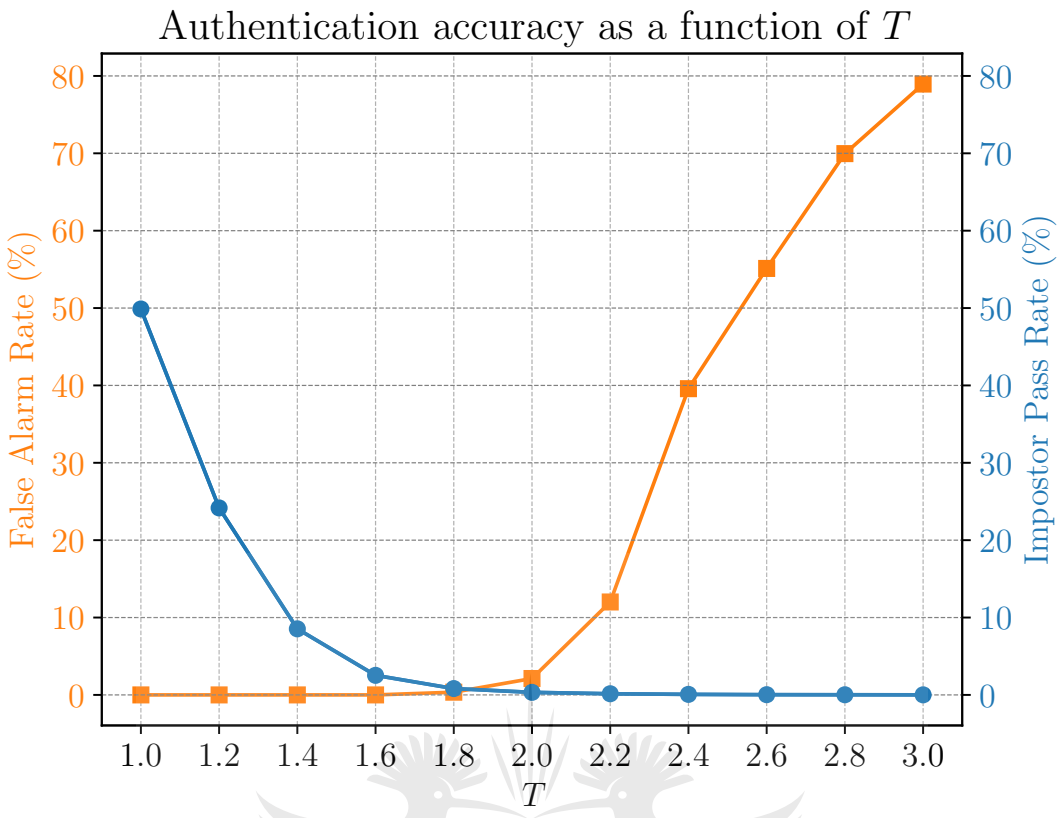


Figure 7.5: Plot shows the trade-off between the FAR and IPR as the decision threshold is varied.

7.2.2 Removal of Common Data

In this experiment, we investigate the effect of the removal of common data on our keystroke analysis algorithm when it is given an authentication task. This this experiment is similar to the one presented in Section 7.1.3, with the difference being the type of task given to the keystroke analysis algorithm.

7.2.2.1 Results

The outcomes of this experiment are presented in Table 7.8 and shown pictorially in Figure 7.6. From the results, it can be observed that the removal of common typing data favours the legal connections test, reason being that as less common data is removed the FAR worsens. The best possible outcomes for this part of the experiment are achieved when $df_{max} \in [0.2, 0.7]$ with a FAR of 0.00% being achieved. The worst FAR is recorded at $df_{max} = 1.0$ where no common data is removed[2]. Unlike in the case of legal connections,

[2]Refer to Table 7.5 to see the approximations of the actual amount of data removed at the various df_{max} values.

the results obtained for the impostor attacks part of the experiment suggest that non-removal of common data has a positive effect on the algorithm's ability to reject impostors. From the plot, it can be seen that the IPR gets better as the amount of common data removed gets less. At the same df_{max} value where the worst FAR was recorded, the best IPR is achieved.

Table 7.8: Experimental results for authentication accuracy at varying stop list sizes.

df_{max}	**Legal Connections**		**Impostor Attacks**	
	False Alarms	**FAR (%)**	**Impostor Passes**	**IPR (%)**
0.1	3	1.06	14542	18.22
0.2	0	0.00	9077	11.37
0.3	0	0.00	6626	8.30
0.4	0	0.00	4883	6.12
0.5	0	0.00	3486	4.37
0.6	0	0.00	2214	2.77
0.7	0	0.00	1274	1.60
0.8	1	0.35	651	0.82
0.9	5	1.77	350	0.44
0.925	7	2.47	280	0.35
0.95	19	6.71	230	0.29
0.975	48	16.96	201	0.25
1.0	85	30.00	154	0.19

As it has been deduced that the removal of common data affects the legal connections and the intrusion tests differently, it is necessary to determine a df_{max} value where the two metrics are at acceptable rates. The point of intersection of the FAR and IPR plots shown in Figure 7.6 was estimated to be approximately at $df_{max} = 0.825$. At this df_{max} value (which respresents the optimal df_{max}), a FAR of 0.35% was achieved due to 1 false rejection and the algorithm granted 571 impostor passes resulting in an IPR of 0.72%. The aforementioned FAR and IPR were obtained by performing the experiment using $df_{max} = 0.825$. This df_{max} value was accepted as the optimal df_{max} setting.

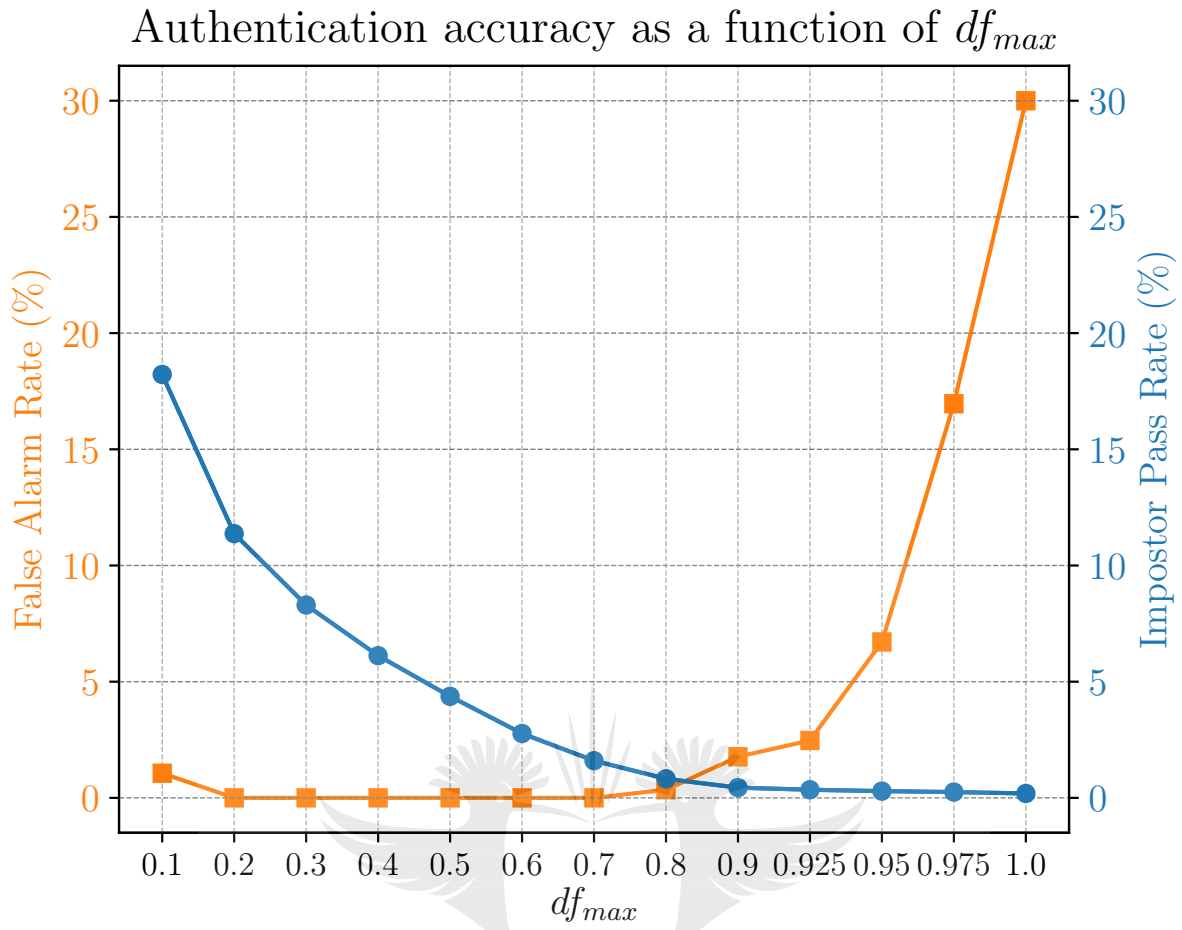


Figure 7.6: Plot shows the trade-off between the FAR and IPR at varying df_{max}.

7.2.3 Query Length

In the experiment detailed in Section 7.1.4, we investigated the effect of the query length on the performance of our keystroke algorithm given a classification task. Therefore, we determined the minimum number of terms (and consequently the number of keystrokes) required to accurately classify a user. We perform a similar experiment here, however, the algorithm is now evaluated on an authentication task with the aim to determine the least number of keystrokes required to perform accurate user verification.

7.2.3.1 Results

Table 7.9 shows the outcomes of the authentication experiments when the query length is varied. Moreover, the results are depicted in Figure 7.7. From the results, it can be easily observed that the algorithm's performance improves with an increase in query length as both the FAR and IPR decrease as $|q|$ increases. The possible explanation for this observation is that as the number of terms in q increases the number of zero components

in $\vec{V}(q)$ reduces thereby improving the effectiveness with which the cosine similarity minimises θ (angle between $\vec{V}(q)$ and $\vec{V}(d)$) in the case of legal connections and maximises θ in the case of impostor attacks. Simply put, as $|q| \to L$ (L is number of terms in the entire vocabulary); $(L - |q|)/L \to 0.0$, therefore, FAR and IPR $\to 0.0\%$[3]. For example, if $|q| = 20$ then $\vec{V}(q)$ will have 20 vector components with $w_{t_i} \neq 0$ and the remaining $L - 20$ components will have $w_{t_i} = 0$. In case of our dataset, $L = 13,264$. Therefore the fraction of vector components with $w_{t_i} = 0$ would be 0.99 (when using $|q| = 20$) which is very substantial.

Regarding the legal connections, acceptable results (FAR $< 1\%$) are obtained from $|q| \geq 300$ with the best possible FAR achieved at $|q| = 400$ and beyond. For query lengths with 400 terms or more, the effect of increasing the number of terms is ineffective. This observation is implied by the saturation of the algorithm as the FAR remains constant at 0.00%.

Table 7.9: Experimental results for authentication accuracy for different query lengths.

$\|q\|$	**Legal Connections**		**Impostor Attacks**	
	False Alarms	**FAR (%)**	**Impostor Passes**	**IPR (%)**
20	50	17.67	5189	6.50
40	30	10.60	2767	3.47
60	14	4.95	1887	2.36
80	11	3.89	1482	1.86
100	11	3.89	1348	1.69
200	3	1.06	911	1.14
300	1	0.35	770	0.96
400	0	0.00	716	0.90
500	0	0.00	646	0.81
600	0	0.00	616	0.77
700	0	0.00	600	0.75
800	0	0.00	578	0.72
900	0	0.00	575	0.72
1000	0	0.00	571	0.72

When looking at the impostor attacks test results, it can be seen that the global minimum

[3]The assumption made when making these approximations is that q is made up of unique terms only– each term appears once and there are no repeating terms. In reality, this is very unlikely, however this assumption makes the discussion presented here less complex.

of the IPR is at approximately $|q| = 1000$. This implies that $|q| = 1000$ is optimal as it reults in the lowest number of impostor passes. However, it is worth noting that the effect of $|q|$ of on impostor passes diminishes as $|q| \to 1000$; this is shown by the algorithm's performance beginning to saturate for approximately $|q| \geq 700$. Therefore, using $|q| \in [700, 1000]$ would result in approximately the same authentication performance as the difference in IPR is not substantial in this range. Prior studies [54, 72] suggest that a typing sample of at least 1000 keystrokes is required in order to achieve adequate authentication performance. However, it can be deduced that acceptable performance (where both the FAR and IPR are less than 1%) is achieved from approximately $|q| \geq 300$. This implies that decent user authentication can be achieved when using a query with as little as 301 keystrokes and if one requires even better performance they can simply increase $|q|$ and consequently the number of required keystrokes.

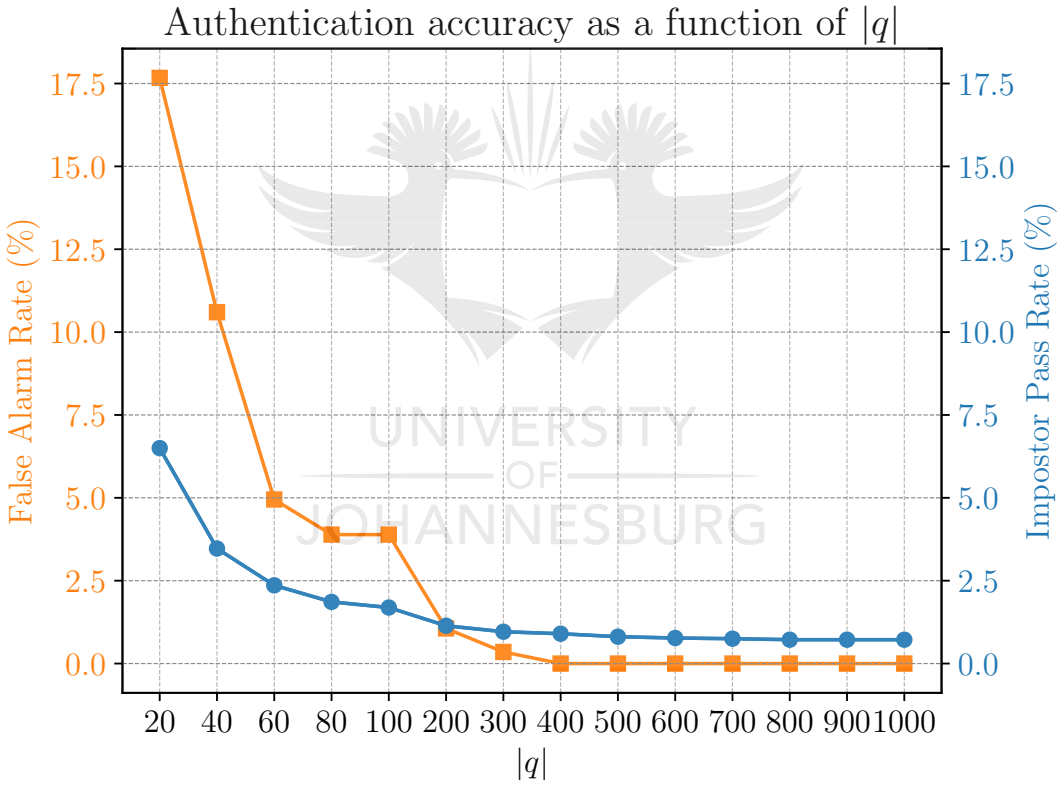


Figure 7.7: Plot shows the trade-off between the FAR and IPR at varying query lengths.

7.2.4 Benchmarking against the State-of-the-art

In this experiment, we benchmark our keystroke dynamics algorithm against one of the state-of-the-art KSD authentication algorithms found in literature. To perform the benchmark, we replicate the free-text based authentication algorithm suggested by Gunetti

and Picardi's [33]. This is achieved by re-implementing their authentication algorithm (in Python) based on the descriptions provided in the paper. Furthermore, the authors were emailed to clarify some details regarding the operation of their algorithm and to obtain original dataset[4] from the published study as it is available upon request.

7.2.4.1 Algorithm

GP's algorithm uses two measures when determining the distance between two typing samples, namely, the "R-Measure" (R stands for "Relative") which was introduced by Bergadano *et al.* [64] and the "A-Measure" (A stands for "Absolute") which was devised by the authors of [33]. The R-Measure is based on the relative duration of n-graphs, i.e., the time between the first and last of n subsequent key-presses. The idea behind the R-Measure is that if the user's typing behaviour is affected by factors such as stress, fatigue or external environmental factors their typing speeds will change uniformly for the typed n-graphs, thereby leaving the relative ordering of the times unchanged. First, the n-graphs are extracted from each typing sample. Thereafter, common n-graphs between the two samples are determined and stored in lists (one for each sample) with the n-graphs sorted by their average duration. In the case where the average digraph are identical, alphabetical order is used as a tie-breaker. The degree of disorder of each list is the ten calculated by taking the sum of the distances between the position of each n-graph in sample 1 and its position in sample 2.

We denote the degree of disorder between two typing samples, S_1 and S_2 as $R_2(S_1, S_2)$ with $n = 2$. R-Measures of different n-graphs can be combined to create a composite. For example, $R_{2,3}(S_1, S_2)$ is defined as the sum of $R_2(S_1, S_2)$ and $R_3(S_1, S_2)$. More formally, R-Measures can be combine as follows[5]:

$$R_{n,m}(S_1, S_2) = R_n(S_1, S_2) + R_m(S_1, S_2) \tag{7.1}$$

The R-Measure suffers from one main limitation: in the case where the two users have the same relative timings but different absolute timings the R-Measure fails as the system may be convinced that the samples come from the same user. To address this limitation, the A-Measure was devised. This measure focuses on the absolute timings of the n-graphs

[4]The dataset used in GP's published study contained typing data from 40 legal users and 165 impostors , the dataset we obtained from GP contained typing samples from 31 legal users and the same number of impostors. The reason for this is that not all the participants consented for their data to be shared with third parties.

[5]The version of (7.1) in the original study is scaled by N and M which are the number of n-graphs and m-graphs shared by samples. However, based on e-mail communication with the study authors, we learnt that the scaling is not necessary. Therefore, we use the non-scaled version of the equation.

to ensure that the typing speeds between two sample under comparison are similar enough to have come from the same user. Two n-graph durations y and z are said to be similar if $1 < max(y, z)/min(y, z) \leq t$; for $t > 1$. We use $t = 1.05$ as this value was found to be optimal in [33][6]. The A-Measure is formally defined as:

$$A_n^t(S_1, S_2) = 1 - (\tau/\phi) \tag{7.2}$$

where τ is the number of similar n-graphs between the S_1 and S_2, and ϕ is the total number of n-graphs shared between S_1 and S_2. Similarly, A-Measures can be combined in the same way as R-Measures and the two measures can be summed.

To authenticate a user given an unknown sample X claimed to belong to a known user A, we first decide on the distance measure, that is, the R-Measure, A-Measure or a combination of the two. The intra-mean distance, $m(A)$, which is the distance between A's reference samples is determined. Thereafter, we compute the inter-mean, $md(B, X)$, distance between X and all other users in the system, B. The unknown sample X is deemed to belong to user A if the following conditions hold [33]:

1. $md(A, X) < md(B, X)$ for all known users in the system other than A;
2. $md(A, X)$ is smaller than $m(A)$ and closer to $m(A)$ than it is to any other $md(B, X)$ value. That is, the following conditions apply:
   (a) $md(A, X) < m(A)$
   (b) $md(A, X) < 0.5[md(B, X) + m(A)]$

7.2.4.2 Experiment

In order to compare of the two methods (ours and GP's), we test our method on GP's dataset and GP's algorithm on one of our datasets, i.e. the dataset collected on the practice site. To apply our algorithm on GP's dataset, we combine all the 15 typing samples provided by each legal user into one large sample. We then perform legal connections and intrusion tests as described in the preceding sections. For the intrusion test, an additional 165 samples (which are impostor samples in GP's dataset) are used to attack the legals users which results in $31 \times (30 + 165) = 6,045$ attempted attacks. In order to evaluate GP's algorithm on our dataset, we split each user's typing profile into 15 samples of 800 keystrokes each[7]. The legal connections and intrusion tests are performed as described in

[6]Based on e-mail communication with the study authors, we learnt that the best t value (for the A-Measure) is 1.05 and not 1.25 as reported in the published study.

[7]In GP's dataset each typing sample consists about 700 to 900 keystrokes, therefore, we create the samples with 800 keystrokes which is the average.

[33]. Furthermore, as an attempt to make the comparison fair, we only use typing data from the top 31 participants (in our dataset) as the dataset received from GP has typing data from 31 legal users.

To tune our algorithm, we use the knowledge learnt about the behaviour of the algorithm from the experiments presented in the preceding sections. Therefore, in this experiment, we use $T = 1.8$, $|q| = 1000$ and $df_{max} = 0.825$ for our algorithm when tested on our dataset. Since the dataset from GP is not the same as the one used in the published study, we perform experiments to determine the optimal distance measure to tune GP's algorithm for their dataset. Moreover, we perform experiments to find the optimal distance measure for our dataset when applying their algorithm to it. Similarly, we conduct an experiment to determine T and df_{max} for when evaluating our algorithm on GP's dataset; we do not perform an experiment to determine $|q|$ instead we use $|q| = 1000$.

7.2.4.3 Results

The experiments described in [33] were replicated, however, the results achieved when applying our re-implementation on their dataset were in some cases far from the results reported in their paper. With our re-implementation, the best results achieved were a FAR of 19.78% and an IPR of 0.2%. These results were achieved using distance measure $R_{2,3} + A_{2,4}$. The IPR achieved was comparable to the one reported in [33], however, the FAR was very far from the FAR reported in the original study. Even after consulting with the study authors and other parties that replicated the algorithm, the FAR did not improve much. Nonetheless, we apply our algorithm on GP's dataset and compare the results with the those reported in [33].

Through experimentation, the optimal parameters when applying our algorithm on GP's dataset were found to be: $df_{max} = 0.4$ and $T = 3.1$. The LUQ and PIQ distributions of GP's dataset obtained using the aforementioned parameters are shown in Figure 7.8. The least overlap (of 18 users) between the two histograms was found to be at $df_{max} = 0.4$. It can be easily noted that the df_{max} optimal for GP's datasets is lesser than that of our dataset. This implies that more stop words are removed from GP's datasets compared to our dataset.

The best results obtained in [33] together with the results achieved by applying our algorithm on GP's dataset are shown Table 7.10. Due to the difference in operation between GP's algorithm and ours, and the dataset size received from GP; the total number of legal connections and attempted intrusions are not the same. In [33], a total of $40 \times 15 = 600$

legal connections and $600 \times 39 \times 15 + 165 \times 40 \times 15 = 450,000$ attacks were attempted. When using our algorithm, a total of 31 legal connections and $31 \times (30 + 165) = 6,045$ intrusions were attempted. Due to these differences we only report the percentages in Table 7.10.

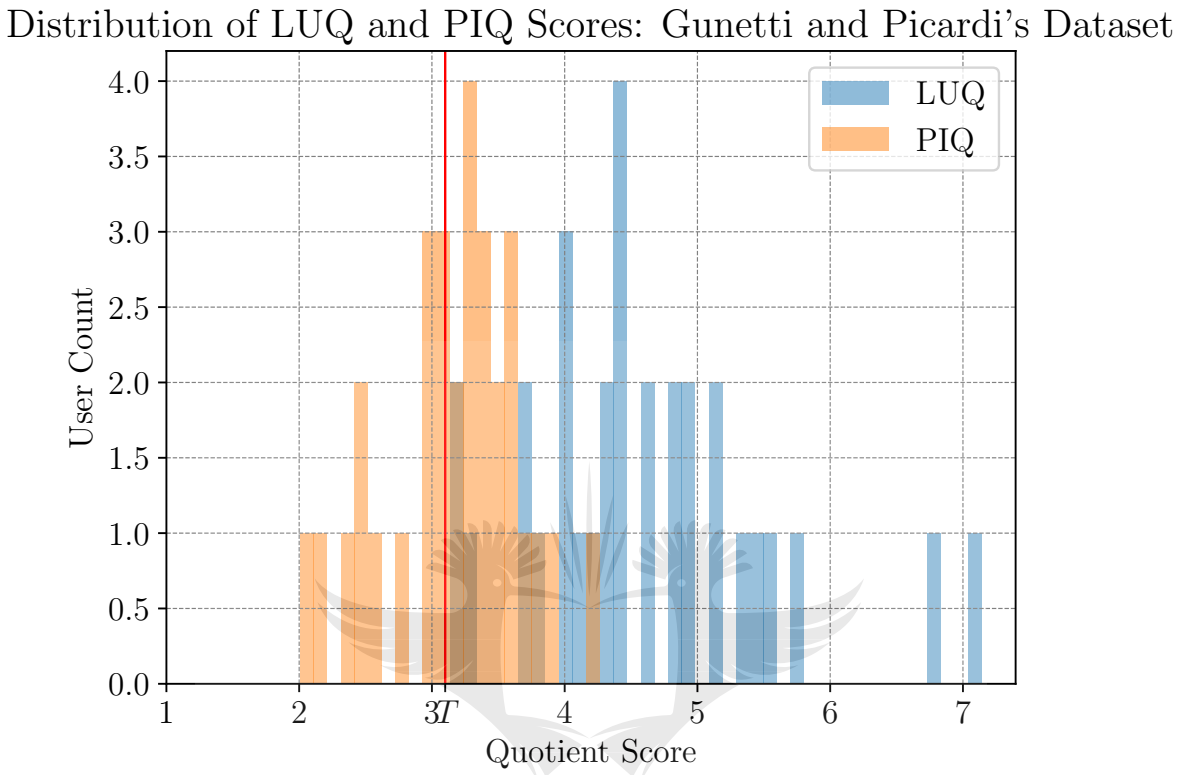


Figure 7.8: Histogram plot showing the distribution of LUQ and PIQ scores for all users in GP's dataset.

From the results presented in Table 7.10, it can be seen that difference between the FARs achieved by both algorithms is not substantial. On the contrary, the IPR achieved with our algorithm is the worst compared to the one in GP's study. The possible explanation for this observation may be overlap of 18 users between the LUQ and PIQ histograms. The overlap is quite high considering that the dataset only consists of 31 users thus resulting in an overlap of $18/31 = 58\%$. Attempts of reducing the IPR by moving the threshold T resulted in increased false rejections thereby increasing the FAR. Therefore, the results shown here are the best possible results achievable when applying our authentication algorithm on GP's dataset.

In terms of the time taken to authenticate a single user, our algorithm out-performed GP's algorithm as the average time taken to authenticate a single user in this experiment was approximately 1.00 second on an Intel Core i7 at 3.0 GHz. Even though the dataset

here was smaller compared to that used in [33] and we had a more faster CPU, we reckon that our algorithm would still be faster even if the experimental environment was exactly as in [33]. We did not have access to a Pentium IV at 2.5 GHz machine (CPU used in [33]) to test our algorithm and see how it would compare.

Table 7.10: Experimental results for authentication accuracy on achieved GP's dataset when apply their algorithm and our text-retrieval based algorithm.

Algorithm	**FAR (%)**	**IPR (%)**
Gunetti & Picardi	3.17	0.03
Ours	3.23	2.22

7.3 Remarks

While the algorithm is clearly capable of authenticating users, and does so reasonably well, it is not without its downfalls.

1. Ideally, a user verification system should only require and use data from the user undergoing verification. Therefore, negative data (data from potential impostors) should be unnecessary as there should be no need to compare to other people in order to verify a user's identity. A serious drawback with the proposed authentication algorithm is that it requires negative data as it compares users when performing authentication. Even though this may work for the number of users we have in the datasets we use, it may not scale well for systems with millions/billions of users as the comparison of typing samples would be time consuming and computationally expensive. Moreover, in such systems (where there is a very high number of users) it is highly likely that the number of users with similar typing traits will increase; therefore using the current authentication rule would most likely increase both the FAR and IPR to a point where the authentication algorithm may become useless.

2. The time needed to build a vocabulary of terms is proportional to the number of keystrokes in a datasets, therefore making the process time consuming for typist datasets of substantial size.

3. Computing the match between two typing samples (using the cosine similarity) is done almost immediately since matches are pre-computed (during the quantisation) when building the vocabulary of terms. Therefore, it is not much of a challenge that a substantial amount of time is spent building a vocabulary as the benefit is that document comparisons can be computed very quickly.

4. Creating a stop list is less computationally expensive compared to the OC-SVM based common data removal method. The reason for this is that a vocabulary is built once, however, with the OC-SVM a new model has to be built every time when the value of ν is changed. Moreover, data removal by means of a stop list is more efficient and provides better accuracy compared to the method suggested in Chapter 5. Therefore, this method address all the limitations of the OC-SVM based method which are discussed in Section 5.5.

7.4 Conclusion

This chapter presented the experiments conducted to evaluate the free-text keystroke dynamics authentication proposed in Chapter 6. The performed experiments tested the proposed algorithm using classification and authentication-based tasks. Furthermore, the suggested authentication algorithm was compared with the state-of-the-art in free-text keystroke analysis. The chapter then concluded with a discussion of the strengths and shortcomings of the proposed keystroke dynamics authentication algorithm.

Chapter 8

Conclusions

This dissertation investigates the use of keystroke dynamics as a means of authentication for online persons. As a result, two algorithms were presented, the first one being the data pre-processing algorithm for detecting and removing common behavioural data for typing behaviour exhibited by most users. The second algorithm was a keystroke dynamics authentication algorithm for free-text input.

The first algorithm (presented in Chapter 5) exploited a novelty detection algorithm– the OC-SVM to identify data points that are common amongst users in a typists dataset. This removal of common data points was performed with the aim of improving the classifiers discrimination ability. The algorithm was then applied to the data to create smaller datasets that contained less common data. Through experimentation, it was shown that classifiers discrimination ability improved as a result of the removal of common data. Even though the common data removal algorithm demonstrated effectiveness, it was not without shortcomings, which include:

1. Fitting the models for varying values of ν, i.e. $\nu \in [0.2, 0.8]$ is a memory intensive task that gets worse as the value of ν increases.

2. The accuracy of the algorithm is very dependant on the sample used to create the super user's typing profile. Therefore, the representativeness of the sample relative to the entire dataset has a significant effect on the performance of the algorithm.

The authentication algorithm borrowed concepts from the field of text retrieval. Here, identity verification by keystroke dynamics was redefined as a text retrieval problem by creating the notion of words/terms built from KSD features. By so doing, we redefined authentication by keystrokes as a text retrieval problem where a user is equivalent to a document and a typing sample provided during authentication is equivalent to a query. The algorithm was evaluated on classification-based tasks and it achieved acceptable

results on both our datasets and three additional (third party) datasets with the best percentage of error being 0.00% and the worst being 13.95%. A further investigation was performed to understand the possible reasons that led to the worst classification performance of our algorithm on one of the third party datasets (Stewart's), these reasons are presented in Section 7.1.2.

One of the key findings made from the classification-based experiments (presented in Section 7.1.2) is that the environmental factors have an effect on how one types (especially if they are novice typists like in the case of the majority of the participants in our study). Some of these environmental factors include: the type of environment (whether it is open or closed), the users stress when sitting for assessments and the type of keyboard used. Notwithstanding the aforementioned observations regarding the effect of changing environmental factors a a user's typing signature, we are of the opinion that changing environmental factors may have minimal effects on skilled typists as they a more defined signature which they have mastered over time. Therefore, we conclude that typing behaviour is just like any other human behaviour which is most likely to change given varying circumstances.

Furthermore, the algorithm was also evaluated on authentication-based tasks where it was also compared against the state-of-the-art algorithm found in KSD literature. One of observations to note made from the authentication-based experiments, particularly the experiment presented in Section 7.2.3, is that using our algorithm, adequate authentication performance (FAR and IPR less than 1%) can be achieved using as little as 301 keystrokes. When tuned using the optimal parameter settings i.e. $T = 1.8$, $|q| = 1000$ and $df_{max} = 0.825$, our method achieved a FAR of 0.00% and an IPR of 0.72%.

When comparing our method to the state-of-the-art, it was learnt that our algorithm does not match the state-of-the-art, especially regarding the IPR– the state-of-the-art achieved an IPR of 0.03% and our algorithm came in at 2.22%. The difference between the FAR achieved by the state-of-the-art algorithm and our algorithm was not significant, i.e. $3.23\% - 3.17\% = 0.06\%$. The only part where our algorithm out-performed the state-of-the-art is regarding the time taken to authenticate a user. It is worth noting that even though the algorithm performed below par when compared to the state-of-the-art, the results achieved are acceptable.

In Chapter 5, we presented a OC-SVM based method for detecting and removing common typing patterns which can be applied to the data at the data pre-processing step. The

application of text retrieval methods in the authentication algorithm provided a more efficient way of removing common data through the concept of a stop list. Common data removal by means of a stop list addressed the issues presented by the OC-SVM. Furthermore, it was shown that both common data removal methods provided more or less the same effect in terms of the amount of data removed and the improvement of the classifiers discrimination ability, however, the stop list proved to be the superior option for the task. One of the benefits of the application of text retrieval methods in KSD is that we were able to build an identity verification system that provides us with an additional ability of easily removing commonly used features.

8.1 Summary of Contributions

The contributions made by this dissertation are summarised below.

1. Two typist datasets of anonymous users collected in real-world situations in two different environments– closed and open-setting environments, i.e. the assessment site and practice site, respectively. Moreover, the collected datasets allow for thorough evaluation of free-text keystroke analysis algorithms as they are sufficiently large.

2. Two methods of detecting and removing common keystroke behaviour which proved to be effective in improving the discrimination of a keystroke analysis classifier. Additionally, the removal of less informative typing data provided side benefits such as reduced model training times and smaller datasets (and smaller index files in the case of the text retrieval based method) that occupy less disk space thereby using less computing resources.

3. A novel free-text keystroke analysis authentication algorithm that borrows concepts and methods from the field of text retrieval. The algorithm's performance is on par with the state-of-the-art authentication algorithms in KSD literature. Our algorithm's performance is on par with the state-of-the-art authentication algorithms achieving similar FAR scores, slightly higher IPR scores but faster execution times.

8.2 Future Work

This section highlights some of the promising avenues of future work that were identified during the course of this study.

With regard to the OC-SVM common data removal method (presented in Chapter 5),

the future may include an investigation into how typing patterns would compare if participants are given a mix of essay and programming typing tasks. Additionally, an investigation into the optimal data removal rate for this dataset of 40% is warranted to see how it may change if:

- different dataset is used;
- the typing tasks are performed by non-skilled vs. skilled typists, or,
- the keystrokes are collected on coding vs. essay-type typing tasks.

Lastly, considering that this algorithm performance relies heavily on how well the data used to create the super user is sampled. It may worthwhile to investigate sampling techniques that would allow for the sampled data to be representative of the whole dataset in order to improve the performance of the the algorithm.

Regarding the proposed text retrieval-based authentication algorithm, it may be interesting to evaluate this method on the task of user identification. The number of clusters (k) for the K-means clustering was chosen empirically. Therefore, it may be beneficial to investigate the optimal k that provides the best quantisation and consequently improve the quality of retrieval. Furthermore, the vocabulary can also be expanded by including larger n-graphs such as trigraphs and four-graphs to create more a complex vocabulary that consist of a mix terms of variable lengths. The effects of environmental conditions on a users typing may be further investigated by performing well designed experiments where variables are controlled and monitored accordingly so that the experiments are more deterministic and more reliable.

The suggested authentication rule suffers from one main drawback (which has been extensively discussed in Section 7.3), i.e. the authentication rule used negative data when authenticating a user. To address this limitation a verification system that does not require negative data can be devised with the aid of one-class classifiers. When using the OCC approach, the authentication decision depends on the data from a single user– the user who is being authenticated. The authors of [81] propose a variation of the OC-SVM that works with textual features. Therefore, it may be advantageous to employing one-class classifier such as the one by [81] instead of using the current authentication rule.

References

[1] A. Gupta, A. Khanna, A. Jagetia, D. Sharma, S. Alekh, and V. Choudhary, "Combining keystroke dynamics and face recognition for user verification," in *Computational Science and Engineering (CSE), 2015 IEEE 18th International Conference on*. IEEE, 2015, pp. 294–299.

[2] M. Radović-Marković *et al.*, "Advantages and disadvantages of e-learning in comparison to traditional forms of learning," *OF THE UNIVERSITY OF PETROŞANI˜ ECONOMICS˜*, p. 289, 2010.

[3] Moodle, "Official Moodle - robust, secure and integrated e-learning platform website, statistics of usage," https://moodle.net/stats/, [Online; accessed 12-Jan-2018].

[4] R. M. Subramanian, "The role of e-learning, the advantages and disadvantages of its adoption in higher education." *Copyright@ 2016 by HINDCO Publications, Tirunelveli-627010*, p. 271, 2016.

[5] F. Monrose and A. D. Rubin, "Keystroke dynamics as a biometric for authentication," *Future Generation computer systems*, vol. 16, no. 4, pp. 351–359, 2000.

[6] J. C. Stewart, J. V. Monaco, S.-H. Cha, and C. C. Tappert, "An investigation of keystroke and stylometry traits for authenticating online test takers," in *2011 International Joint Conference on Biometrics (IJCB)*. IEEE, 2011, pp. 1–7.

[7] A. Maas, C. Heather, C. Do, R. Brandman, D. Koller, and A. Ng, "Offering verified credentials in massive open online courses: Moocs and technology to advance learning and learning research (ubiquity symposium)," *Ubiquity*, vol. 2014, no. May, pp. 1–11, 2014.

[8] P. S. Teh, A. B. J. Teoh, and S. Yue, "A survey of keystroke dynamics biometrics," *The Scientific World Journal*, vol. 2013, 2013.

[9] E. Alsolami, "An examination of keystroke dynamics for continuous user authentication," Ph.D. dissertation, Queensland University of Technology, 2012.

[10] J. D. Woodward, N. M. Orlans, and P. T. Higgins, *Biometrics:[identity assurance in the information age]*. McGraw-Hill/Osborne New York, 2003.

[11] W. E. Cooper, *Cognitive aspects of skilled typewriting*. Springer Science & Business Media, 2012.

[12] D. Hochfelder, *The telegraph in America, 1832–1920*. JHU Press, 2012.

[13] K. Beauchamp, *History of telegraphy*. Iet, 2001, no. 26.

[14] C. L. Sholes, "Typewriting machine," United States Patent 207,559, 1878, August.

[15] C. L. Sholes, C. Glidden, and S. Soule, "Typewriting machine," United States Patent 79,868, 1868, July.

[16] A. Dvorak, N. L. Merrick, W. L. Dealey, and G. C. Ford, "Typewriting behavior," *New York: American Book Company*, 1936.

[17] P. Gregory and M. A. Simon, *Biometrics for dummies*. John Wiley & Sons, 2008.

[18] R. S. Gaines, W. Lisowski, S. J. Press, and N. Shapiro, "Authentication by keystroke timing: Some preliminary results," Rand Corp Santa Monica CA, Tech. Rep., 1980.

[19] A. K. Jain, P. Flynn, and A. A. Ross, *Handbook of biometrics*. Springer Science & Business Media, 2007.

[20] W. Stallings, L. Brown, M. D. Bauer, and A. K. Bhattacharjee, *Computer security: principles and practice*. Pearson Education, 2012.

[21] R. E. Smith, *Authentication: from passwords to public keys*. Addison-Wesley Longman Publishing Co., Inc., 2001.

[22] H. Barghouthi, "Keystroke dynamics. how typing characteristics differ from one application to another," Master's thesis, Gjvik University College, 2009.

[23] P. H. Pisani, A. C. Lorena, and A. C. de Carvalho, "Adaptive approaches for keystroke dynamics," in *Neural Networks (IJCNN), 2015 International Joint Conference on*. IEEE, 2015, pp. 1–8.

[24] T. Highlander, D. Bassett, and D. Boone, "Utilization of keyboard dynamics for unique identification of human users," in *Aerospace and Electronics Conference, NAECON 2014-IEEE National*. IEEE, 2014, pp. 153–156.

[25] K. Revett, *Behavioral biometrics: a remote access approach*. John Wiley & Sons, 2008.

[26] A. K. Jain, R. Bolle, and S. Pankanti, *Biometrics: personal identification in networked society*. Springer Science & Business Media, 2006, vol. 479.

[27] K. Hempstalk, "Continuous typist verification using machine learning," Ph.D. dissertation, The University of Waikato, 2009.

[28] S. Das and J. Debbarma, "Designing a biometric strategy (fingerprint) measure for enhancing atm security in indian e-banking system," *International Journal of Information and Communication Technology Research*, vol. 1, no. 5, 2011.

[29] L. Coventry, A. De Angeli, and G. Johnson, "Usability and biometric verification at the atm interface," in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 2003, pp. 153–160.

[30] M. Trojahn and F. Ortmeier, "Biometric authentication through a virtual keyboard for smartphones," *International Journal of Computer Science & Information Technology*, vol. 4, no. 5, p. 1, 2012.

[31] D. Shanmugapriya and G. Padmavathi, "Virtual key force–a new feature for keystroke," *J. International Journal of Engineering Science and Technology*, vol. 3, no. 10, pp. 738–743, 2011.

[32] M. Nisenson, I. Yariv, R. El-Yaniv, and R. Meir, "Towards behaviometric security systems: Learning to identify a typist," in *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 2003, pp. 363–374.

[33] D. Gunetti and C. Picardi, "Keystroke analysis of free text," *ACM Transactions on Information and System Security (TISSEC)*, vol. 8, no. 3, pp. 312–347, 2005.

[34] J. C. Stewart, J. V. Monaco, S.-H. Cha, and C. C. Tappert, "An investigation of keystroke and stylometry traits for authenticating online test takers," in *Biometrics (IJCB), 2011 International Joint Conference on*. IEEE, 2011, pp. 1–7.

[35] A. A. Ahmed and I. Traore, "Biometric recognition based on free-text keystroke dynamics," *IEEE transactions on cybernetics*, vol. 44, no. 4, pp. 458–472, 2014.

[36] H. Çeker and S. Upadhyaya, "User authentication with keystroke dynamics in long-text data," in *Biometrics Theory, Applications and Systems (BTAS), 2016 IEEE 8th International Conference on*. IEEE, 2016, pp. 1–6.

[37] R. Janakiraman and T. Sim, "Keystroke dynamics in a general setting," in *International Conference on Biometrics*. Springer, 2007, pp. 584–593.

[38] C. Murphy, J. Huang, D. Hou, and S. Schuckers, "Shared dataset on natural human-computer interaction to support continuous authentication research," in *2017 IEEE International Joint Conference on Biometrics (IJCB)*. IEEE, 2017, pp. 525–530.

[39] P. S. Dowland and S. M. Furnell, "A long-term trial of keystroke profiling using digraph, trigraph and keyword latencies," in *IFIP International Information Security Conference*. Springer, 2004, pp. 275–289.

[40] J. C. Rodríguez del Pino, E. Rubio Royo, and Z. J. Hernández Figueroa, "Vpl: laboratorio virtual de programación para moodle," in *XVI Jornadas de Enseñanza Universitaria de la Informática*. Universidade de Santiago de Compostela. Escola Técnica Superior d'Enxeñaría, 2010, pp. 429–435.

[41] Ace, "Ace Editor," https://ace.c9.io/, [Online; accessed 05-Jun-2018].

[42] J. Noyes, "Qwerty-the immortal keyboard," *Computing & Control Engineering Journal*, vol. 9, no. 3, pp. 117–122, 1998.

[43] C. C. Aggarwal, "Outlier analysis," in *Data mining*. Springer, 2017.

[44] P. Juszczak, D. Tax, and R. P. Duin, "Feature scaling in support vector data description," in *Proc. ASCI*. Citeseer, 2002, pp. 95–102.

[45] I. Jolliffe, *Principal component analysis*. Springer, 2011.

[46] S. Raschka and V. Mirjalili, *Python machine learning*. Packt Publishing Ltd, 2017.

[47] T. Mokoena and D. Sabatta, "Identifying and removing common behaviour in keystroke dynamics to improve classification," in *Proc. SATNAC*. SATNAC, 2019.

[48] C. E. Shannon, "A mathematical theory of communication," *Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[49] C. Manning, P. Raghavan, and H. Schütze, "Introduction to information retrieval," *Natural Language Engineering*, vol. 16, no. 1, pp. 100–103, 2010.

[50] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[51] A. Bounsiar and M. G. Madden, "Kernels for one-class support vector machines," in *2014 International Conference on Information Science & Applications (ICISA)*. IEEE, 2014, pp. 1–4.

[52] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[53] T. Mokoena and D. Sabatta, "User classification by keystroke dynamics using text retrieval methods," in *Proc. PRASA*. SAUPEC/RobMech/PRASA International Conference, 2020.

[54] D. Gunetti, C. Picardi, and G. Ruffo, "Keystroke analysis of different languages: a case study," in *International Symposium on Intelligent Data Analysis*. Springer, 2005, pp. 133–144.

[55] J. Huang, D. Hou, S. Schuckers, T. Law, and A. Sherwin, "Benchmarking keystroke authentication algorithms," in *Information Forensics and Security (WIFS), 2017 IEEE Workshop on*. IEEE, 2017, pp. 1–6.

[56] R. Joyce and G. Gupta, "Identity authentication based on keystroke latencies," *Communications of the ACM*, vol. 33, no. 2, pp. 168–176, 1990.

[57] S. Bleha, C. Slivinsky, and B. Hussien, "Computer-access security systems using keystroke dynamics," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 12, no. 12, pp. 1217–1222, 1990.

[58] K. Revett, S. T. De Magalhães, and H. M. Santos, "Enhancing login security through the use of keystroke input dynamics," in *International Conference on Biometrics*. Springer, 2006, pp. 661–667.

[59] E. Yu and S. Cho, "Novelty detection approach for keystroke dynamics identity verification," in *International conference on intelligent data engineering and automated learning*. Springer, 2003, pp. 1016–1023.

[60] R. N. Rodrigues, G. F. Yared, C. R. d. N. Costa, J. B. Yabu-Uti, F. Violaro, and L. L. Ling, "Biometric access control through numerical keyboards based on keystroke dynamics," in *International Conference on Biometrics*. Springer, 2006, pp. 640–646.

[61] M. Brown and S. J. Rogers, "User identification via keystroke characteristics of typed names using neural networks," *International Journal of Man-Machine Studies*, vol. 39, no. 6, pp. 999–1014, 1993.

[62] L. K. Maisuria, C. S. Ong, and W. K. Lai, "A comparison of artificial neural networks and cluster analysis for typing biometrics authentication," in *IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No. 99CH36339)*, vol. 5. IEEE, 1999, pp. 3295–3299.

[63] D. Umphress and G. Williams, "Identity verification through keyboard characteristics," *International journal of man-machine studies*, vol. 23, no. 3, pp. 263–273, 1985.

[64] F. Bergadano, D. Gunetti, and C. Picardi, "User authentication through keystroke dynamics," *ACM Transactions on Information and System Security (TISSEC)*, vol. 5, no. 4, pp. 367–397, 2002.

[65] D. Gunetti, C. Picardi, and G. Ruffo, "Dealing with different languages and old profiles in keystroke analysis of free text," in *Congress of the Italian Association for Artificial Intelligence*. Springer, 2005, pp. 347–358.

[66] P. Dowland, S. Furnell, and M. Papadaki, "Keystroke analysis as a method of advanced user authentication and response," in *Security in the Information Society*. Springer, 2002, pp. 215–226.

[67] A. Messerman, T. Mustafić, S. A. Camtepe, and S. Albayrak, "Continuous and non-intrusive identity verification in real-time environments based on free-text keystroke dynamics," in *Biometrics (IJCB), 2011 International Joint Conference on*. IEEE, 2011, pp. 1–8.

[68] P. Dowland, "A preliminary investigation of user authentication using continuous keystroke analysis," in *Proc IFIP Annual Working Conf on Information Security Management and Small System Security, 2001*, 2001, pp. 27–28.

[69] H. Davoudi and E. Kabir, "A new distance measure for free text keystroke authentication," in *Computer Conference, 2009. CSICC 2009. 14th International CSI*. IEEE, 2009, pp. 570–575.

[70] K. Xi, Y. Tang, and J. Hu, "Correlation keystroke verification scheme for user access control in cloud computing environment," *The Computer Journal*, vol. 54, no. 10, pp. 1632–1644, 2011.

[71] T. Shimshon, R. Moskovitch, L. Rokach, and Y. Elovici, "Continuous verification using keystroke dynamics," in *Computational Intelligence and Security (CIS), 2010 International Conference on*. IEEE, 2010, pp. 411–415.

[72] J. Huang, D. Hou, S. Schuckers, and Z. Hou, "Effect of data size on performance of free-text keystroke authentication," in *Identity, Security and Behavior Analysis (ISBA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1–7.

[73] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *null*. IEEE, 2003, p. 1470.

[74] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2. Ieee, 2006, pp. 2161–2168.

[75] TypingDNA, "TypingDNA," https://www.typingdna.com/, [Online; accessed 05-Jan-2018].

[76] D. Security, "Keystroke Recognition," https://http://www.deepnetsecurity.com/, [Online; accessed 06-Jun-2018].

[77] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information processing & management*, vol. 24, no. 5, pp. 513–523, 1988.

[78] C. C. Tappert, M. Villani, and S.-H. Cha, "Keystroke biometric identification and authentication on long-text input," in *Behavioral biometrics for human identification: Intelligent applications*. IGI global, 2010, pp. 342–367.

[79] J. V. Monaco, N. Bakelman, S.-H. Cha, and C. C. Tappert, "Developing a keystroke biometric system for continual authentication of computer users," in *2012 European Intelligence and Security Informatics Conference*. IEEE, 2012, pp. 210–216.

[80] J. V. Monaco, J. C. Stewart, S.-H. Cha, and C. C. Tappert, "Behavioral biometric verification of student identity in online course assessment and authentication of authors in literary works," in *2013 IEEE Sixth International Conference on Biometrics: Theory, Applications and Systems (BTAS)*. IEEE, 2013, pp. 1–8.

[81] L. M. Manevitz and M. Yousef, "One-class svms for document classification," *Journal of machine Learning research*, vol. 2, no. Dec, pp. 139–154, 2001.

Appendices

A Data Collection Schedule

Below is the schedule that was followed during the data collection phase. The assessments were usually conducted on Thursdays afternoon, with an exception on the 24th and 29th of May. An assessment session being an hour long. Each student was only entitled to no more than a single session per assessment day.

Table A1: Data collection schedule.

Event	**Date**
Publishing of the informed consent form	11 February 2019
Data collection started on practise site	20 February 2019
Assessment 1	14 March 2019
Public holiday (No Assessment)	21 March 2019
Assessment 2	04 April 2019
Assessment 3	11 April 2019
Assessment 4	18 April 2019
Assessment 5	25 April 2019
Assessment 6	02 May 2019
Assessment 7	09 May 2019
Assessment 8	16 May 2019
Assessment 9	24 May 2019
Assessment 10	29 May 2019
Assessment 11	30 May 2019
Assessment 12	6 June 2019
Practice site was taken down	7 June 2019

B Participant Keystroke Contribution

Below is a table of the keystroke contributions made by the 283 participants who consented for their to be recorded and for experimentation for the purpose of this dissertation. The table shows the number of keystrokes recorded from each participant on both the practice and assessment site.

Table B1: Keystroke contributions (on the practice and assessment sites) of the participants that consented.

619	129436	12710
462	118406	13399
556	87879	13172
452	72272	12472
627	67767	11222
568	62865	18925
489	57819	12084
482	57077	10809
394	56995	20081
488	56711	17696
508	56660	8134
320	51717	8542
425	50980	7734
463	50790	11637
588	49549	14240
518	48501	10020
480	46789	6729
348	44283	10368
376	44242	13307
582	43876	10196
457	43575	12436
407	41926	9050
540	41478	13394
436	41305	13485
370	41130	11244
400	40803	10358
346	40647	9283

Continued on the next page

Table B1: Keystroke contributions (on the practice and assessment sites) of the participants that consented.

User ID	Practice Site	Assessment Site
506	40181	16977
605	39230	10693
419	38820	9566
473	38447	8015
481	38353	11323
415	38157	12612
546	37715	11538
460	37695	8455
527	36931	11800
581	36089	13274
592	35550	9277
547	34962	16870
535	33808	8453
580	33464	12150
632	33061	14005
598	32863	5568
538	31449	15147
327	31305	11111
392	31153	11823
607	31147	11254
554	30933	15155
510	30669	13173
429	30665	15695
422	30606	10543
628	30474	12178
549	30468	13232
364	30274	14540
413	29999	13991
477	29984	9081
389	29698	10153
414	28980	15985

Continued on the next page

Table B1: Keystroke contributions (on the practice and assessment sites) of the participants that consented.

User ID	Practice Site	Assessment Site
437	27846	11382
447	27800	14666
620	27752	15016
385	27649	25504
431	27330	8450
523	27261	10205
558	26752	6884
443	26749	8550
626	26680	11899
491	26665	6586
405	26634	10704
375	26415	9861
603	26300	8345
401	26282	10371
593	26172	10541
525	25941	7102
634	25792	10274
574	25652	6703
409	25535	10134
496	25215	16134
432	25180	13277
577	25158	10672
559	25100	10873
542	24702	12050
492	24581	11594
371	24383	10738
601	24220	6478
622	24202	17866
412	23938	9314
534	23857	10075
618	23844	13703

Continued on the next page

Table B1: Keystroke contributions (on the practice and assessment sites) of the participants that consented.

User ID	Practice Site	Assessment Site
587	23792	9466
379	23609	8725
428	23561	8676
456	23547	16150
550	23491	5426
503	23397	9032
541	23362	9022
613	23326	11507
465	23284	11095
529	23260	11169
571	23235	11649
361	23033	8867
380	23027	6204
486	22975	5572
369	22859	4116
345	22652	6327
625	22482	8973
524	22432	9061
490	22398	13502
359	22355	9185
381	22227	13299
471	22219	8307
483	22146	10487
614	21923	15492
404	21715	12208
435	21592	9571
532	21590	9125
537	21559	9818
596	21540	10059
515	21440	11315
338	21139	9458

Continued on the next page

Table B1: Keystroke contributions (on the practice and assessment sites) of the participants that consented.

User ID	Practice Site	Assessment Site
536	20996	12282
585	20936	8456
382	20899	11548
485	20812	9492
406	20552	7675
418	20522	7230
615	20516	13051
557	20500	12486
499	20412	10305
368	20140	9953
495	20091	14153
479	19888	8582
390	19870	11117
325	19815	8973
629	19627	6687
579	19576	11723
563	19529	8291
399	19445	9149
555	19275	8614
484	19260	8316
363	19124	8349
594	19074	10541
528	18993	13242
526	18978	10272
548	18766	11570
475	18727	11405
451	18718	8921
420	18716	16787
468	18594	12819
514	18441	9986
621	18230	6815

Continued on the next page

Table B1: Keystroke contributions (on the practice and assessment sites) of the participants that consented.

User ID	Practice Site	Assessment Site
397	18229	8968
395	18198	10574
464	18133	11227
616	18129	18790
423	18058	11342
570	17994	8167
509	17967	11342
349	17918	6713
478	17816	10256
576	17711	14864
398	17698	8942
396	17582	10676
421	17419	9770
366	17418	11976
591	17388	7054
445	17388	9231
602	17386	12894
362	17220	6180
531	17174	5576
590	16844	1628
606	16779	11095
501	16658	11357
575	16633	8088
565	16552	10491
517	16508	14512
560	16395	9910
360	16297	10576
402	16101	5142
459	15986	16678
566	15546	12604
530	15494	6699

Continued on the next page

Table B1: Keystroke contributions (on the practice and assessment sites) of the participants that consented.

User ID	Practice Site	Assessment Site
440	15478	6808
393	15435	15597
324	15403	11318
504	15378	12412
520	15222	7984
609	15108	9697
533	14924	8207
507	14835	9691
319	14809	1798
539	14757	13620
340	14755	8552
624	14648	6065
446	14628	7219
384	14413	9381
595	14068	8093
355	14061	8367
500	13924	5704
356	13843	5790
586	13754	12932
354	13677	12992
417	13339	13048
635	12893	10733
502	12787	15015
551	12687	5302
350	12498	9436
608	12467	16077
388	12345	12190
467	12035	18653
569	11897	14674
472	11859	7120
410	11667	9086

Continued on the next page

Table B1: Keystroke contributions (on the practice and assessment sites) of the participants that consented.

User ID	Practice Site	Assessment Site
597	11664	9745
426	11641	10154
383	11309	10088
521	11064	9131
589	11003	8258
438	10959	9042
449	10920	9044
553	10907	3801
391	10817	7845
433	10790	11324
599	10657	11506
584	10483	9320
604	10462	8475
583	10450	16600
578	10431	3550
434	10235	13459
617	10218	11651
335	10175	10452
519	10161	9844
573	10151	5954
512	10030	10634
458	9993	6873
474	9829	8628
358	9757	6496
318	9450	7266
367	9404	10097
562	9263	13452
326	9215	4857
623	9175	9036
487	8937	12986
544	8884	10003

Continued on the next page

Table B1: Keystroke contributions (on the practice and assessment sites) of the participants that consented.

User ID	Practice Site	Assessment Site
454	8880	2149
543	8455	11932
411	8387	12057
441	8335	9942
453	8128	5421
610	8075	7618
493	7876	7872
374	7705	6813
439	7442	12142
466	7277	11379
545	7179	7243
372	7118	8701
337	7079	6685
505	7066	3368
323	6974	2725
329	6931	5816
342	6849	6475
564	6773	13164
497	6614	14563
516	6514	6941
511	6442	4198
561	6371	12490
469	6267	12621
373	6212	4495
416	5975	7701
448	5887	13005
328	5748	6009
424	5622	954
427	5390	6334
494	5380	5456
450	5298	7863

Continued on the next page

Table B1: Keystroke contributions (on the practice and assessment sites) of the participants that consented.

User ID	Practice Site	Assessment Site
334	5163	1606
611	5103	4458
357	5078	1441
322	4565	7853
386	4376	3550
377	4118	7307
331	4088	7112
461	3706	2399

C Common Digraphs Removed by the OC-SVM

Table C1: The top 10 digraphs (ordered in descending order) removed by the One-Class Support Vector Machines (OC-SVM) based common data removal algorithm at varying data removal rates.

% of Data Removed	Top 10 Removed Digraphs
20	Backspace Backspace Right-Arrow Right-Arrow in nt Left-Arrow Left-Arrow Right-Arrow Backspace Down-Arrow Down-Arrow ; Enter su t Space
30	Backspace Backspace Right-Arrow Right-Arrow in Left-Arrow Left-Arrow nt Down-Arrow Down-Arrow ; Enter Right-Arrow Backspace um su
40	Backspace Backspace Right-Arrow Right-Arrow in Left-Arrow Left-Arrow nt Down-Arrow Down-Arrow ; Enter Up-Arrow Up-Arrow um Right-Arrow Backspace

50	Backspace Backspace Right-Arrow Right-Arrow in Left-Arrow Left-Arrow Down-Arrow Down-Arrow nt Up-Arrow Up-Arrow ; Enter um t Space
60	Backspace Backspace Right-Arrow Right-Arrow Left-Arrow Left-Arrow in Down-Arrow Down-Arrow nt Up-Arrow Up-Arrow ; Enter t Space um
70	Backspace Backspace Right-Arrow Right-Arrow Left-Arrow Left-Arrow in Down-Arrow Down-Arrow nt Up-Arrow Up-Arrow ; Enter t Space Enter Enter

80	Backspace Backspace Right-Arrow Right-Arrow Left-Arrow Left-Arrow in Down-Arrow Down-Arrow nt Up-Arrow Up-Arrow ; Enter t Space Enter Enter